HELSINGIN KAUPPAKORKEAKOULU
Liiketoiminnan teknologian laitos

GENEETTINEN ALGORITMI KULJETUSOPTIMOINTIONGELMAAN

Liikkeenjohdon systeemien
pro gradu -tutkielma
Anna Metsäranta
kevät 2005

Liiketoiminnan teknologian laitoksen

laitosneuvoston kokouksessa 28/2 2005 hyväksytty

arvosanalla Erinomainen, 90 p.

Markku Kallio KTT Merja Halme KTT

HELSINGIN KAUPPAKORKEAKOULU
Liiketoiminnan teknologian laitos
Liikkeenjohdon systeemien pro gradu –tutkielma
Anna Metsäranta 70751-2

TIIVISTELMÄ

09.02.2005

GENEETTINEN ALGORITMI KULJETUSOPTIMOINTIONGELMAAN

Tavoitteet

Tämän tutkielman tavoitteena on optimoida realistinen kuljetusongelma geneettisellä algoritmilla. Esimerkkiongelmana käytetään raakapuun laivausta Itämerellä. Tutkimuksen pääasiallinen tarkoitus on arvioida geneettisen algoritmin soveltuvuutta monimutkaisen kuljetusongelman optimointiin, missä ratkaisuvaihtoehdoille asetetaan useita eri tyyppisiä rajoitteita. Lisäksi tarkoituksena on tutkia operaattoreiden ja parametrien arvojen vaikutusta optimointiin.

Tutkimusmenetelmät

Puunlaivausongelmasta muodostetaan matemaattinen malli, jossa minimoidaan kuljetuskustannuksia. Ongelman optimointia varten ohjelmoidaan geneettinen algoritmi C-ohjelmointikielellä. Optimoinnissa kokeillaan useita erilaisia operaattoreita ja parametrien arvoja, joille pyritään löytämään optimaalinen kombinaatio siten, että geneettisellä algoritmilla saadaan mahdollisimman hyvä ratkaisu kohtuullisessa ajassa. Tulosten arviointia varten puunkuljetusongelma ratkaistaan myös GAMS-optimointiohjelmistolla.

Tulokset

Geneettinen algoritmi ei pärjää ratkaisun laadussa ja suoritustehossa GAMS:lle. Algoritmilla saatu tulos on kuitenkin kohtuullinen, ja geneettisen algoritmin etu on mahdollisuus tehdä ongelmasta realistinen. Parametrien arvoilla havaittiin olevan selvä vaikutus algoritmin toimintaan.

Tämän tyyppisessä, separoitumattomassa ongelmassa geneettisen algoritmin toimintaperiaatetta oli vaikea hyödyntää. On kuitenkin syytä uskoa, että geneettinen algoritmi olisi varteenotettava ratkaisukeino suuriinkin separoituviin kuljetusongelmiin.

Avainsanat: geneettiset algoritmit, kuljetusten optimointi

SISÄLLYSLUETTELO

1 JOHDANTO

Monet yrityselämän päätösongelmat ovat niin monimutkaisia, että niiden ratkaiseminen yksinkertaisilla menetelmillä kuten lineaarisella optimoinnilla tai enumeroimalla olisi mahdotonta. Tällaisten ongelmien ratkaisemiseksi on kehitetty useita erilaisia heuristiikkoja eli kokeillen etsiviä metodeita. Eräs tällaisten menetelmien perhe on evoluutioalgoritmit. Niissä muodostetaan ongelman mahdollisista ratkaisuista populaatio, jota muokataan luonnossa esiintyvien operaatioiden tavoin. Evoluutiolaskennan piiriin kuuluvat geneettiset algoritmit, jotka ovat joukko luonnon evoluutioon perustuvia periaatteita. Geneettiset algoritmit ovat stokastisia heuristiikkoja, eli hakuun sisältyy satunnaistekijöitä. Niillä on ratkaistu tehokkaasti lähinnä kombinatorisia ongelmia, joissa käypien ratkaisujen joukko on diskreetti ja joihin usein liittyy ns. kombinatorinen räjähdys eli ratkaisuavaruuden kasvaminen kiihtyvällä vauhdilla päätösmuuttujien määrän kasvaessa. Tällaisten ongelmien ratkaisu yksinkertaisemmilla menetelmillä on usein vaikeaa tai mahdotonta.

Geneettisissä algoritmeissa evoluutiobiologian periaatteet yhdistyvät ongelmanratkaisuun informaatioteknologian avulla. Menetelmän periaatteita noudattaen voidaan kulloinkin kyseessä olevaan ongelmaan räätälöidä sopiva ratkaisualgoritmi. Geneettisiä algoritmeja on käytetty sekä kokonais- että liukulukuja sisältävien ongelmien ratkaisussa. Nykyään käytetään usein puhtaan geneettisen algoritmin sijasta hybridiä, jossa perusalgoritmin toimintaa on tehostettu jollakin ongelmaan liittyvällä erityisosaamisella.

Alan kulmakivenä pidetään John Hollandin teosta *Adaptation in Natural and Artificial Systems* (The University of Michigan Press, Ann Arbor, 1975). Geneettisiä algoritmeja on tutkittu paljon parin viimeisen vuosikymmenen aikana ja sovellettu menestyksekkäästi useilla tekniikan, tieteen ja kaupan aloilla. Biologian periaatteiden jäljittely tekee geneettisistä algoritmeista erityisen kiinnostavia, sillä luonnon mekanismien ja evoluution menetelmien syvempi ymmärtäminen avaavat jatkuvasti uusia mahdollisuuksia tehokkaampien adaptiivisten ratkaisumenetelmien kehittämiseen.

Geneettisiä algoritmeja on usein käytetty melko kevyesti strukturoitujen ongelmien ratkaisemisessa, joissa ei ole asetettu monia rajoituksia ratkaisukandidaattien käypyydelle. Tässä tutkimuksessa ohjelmoidaan geneettinen algoritmi optimoimaan raakapuunkuljetuksia Itämerellä. Ongelmanasettelu on realistinen ja ottaa huomioon mahdollisimman monta aidossa päätöstilanteessa mukana olevaa tekijää. Tutkimuksessa käytetty aineisto on muokattu Saara Lallukan pro gradu –tutkimuksessaan *Mixed Integer Programming in the Selection of Roundwood Vessels, case Thomesto* (2003) käyttämästä aineistosta. Tämän tutkimuksen tarkoituksena on tarkastella geneettisen algoritmin soveltuvuutta monimutkaisen reaalielämän ongelman ratkaisuun. Algoritmi ohjelmoidaan C-ohjelmointikielellä. Tulosten arviointia ja vertailua varten ratkaistaan ongelma myös GAMS-optimointiohjelmistolla (*the General Algebraic Modeling System*).

Tutkielma on jaettu seitsemään lukuun. Seuraavassa luvussa esitetään geneettisten algoritmien teoriaa. Luvuissa 3 ja 4 esitellään optimoitava puunkuljetusongelma sekä sen matemaattinen malli. Luku 5 käsittelee ongelman optimointiin ohjelmoitua geneettistä algoritmia. Luvussa 6 esitellään tutkimuksen tulokset ja luvussa 7 pohditaan tulosten merkitystä.

2 GENEETTISTEN ALGORITMIEN TEORIAA

2.1 TAUSTAA

Evoluutio tarkoittaa lajien jatkuvaa kehittymistä muuttuvaan ympäristöön sopeutumisen seurauksena. Luonnon evoluutio on hidasta mutta jatkuvaa oppimista, jonka työkaluina toimivat valinta, lisääntyminen, mutaatio ja kilpailu. 1950- ja 60-luvuilla monet tietojenkäsittelyopin tutkijat kehittelivät ajatuksia evoluutiosta optimointityökaluna tekniikan alan ongelmissa. Evoluution mekanismeissa nähtiin mahdollisuus kehittää adaptiivinen eli vaihtuvissa olosuhteissa hyvin toimiva ratkaisumenetelmä stokastisiin ongelmiin. Ideana oli muodostaa populaatio ongelman mahdollisista ratkaisuista ja käyttää sitten luonnon evoluutiosta lainattuja operaatioita populaation muokkaamiseen. Voidaan puhua jopa evoluution simuloimisesta tietokoneella (Simula 1997).

John Holland kehitti geneettisten algoritmien perusidean 1960-luvulla. Hän työsti ajatustaan Michiganin yliopistossa yhdessä oppilaidensa kanssa 1960- ja 1970-luvuilla ja julkaisi ensimmäisen teoksensa geneettisistä algoritmeista vuonna 1975. Hollandin alkuperäinen tavoite ei ollut suunnitella algoritmeja ratkaisemaan joitakin tiettyjä ongelmia, vaan hän halusi tutkia luonnossa esiintyvää sopeutumista ja kehitellä tapoja, joilla tämän sopeutumisen voisi tuoda tietokonejärjestelmiin. Hollandin populaatioihin perustuva algoritmi, joka käytti operaatioinaan risteytystä ja mutaatiota, oli merkittävä innovaatio. Nykyään geneettinen algoritmi on hyvin joustava käsite ja monet sovellukset, joista käytetään tätä nimeä, ovat kaukana Hollandin alkuperäisestä käsityksestä. (Mitchell 1996, 2-3)

2.1.1 Termistö

Geneettisten algoritmien yhteydessä käytetty termistö on lainattu biologiasta ja se on alalla vakiintunut. *Yksilö* tarkoittaa jonkin mahdollisen ratkaisun ohjelmoitua muotoa eli ratkaisuavaruuden pistettä. Samassa merkityksessä käytetään termejä *alkio* ja *yrite*. Yksilö koostuu *kromosomeista*, jotka ovat merkkijonoja. Kromosomit puolestaan

voidaan jakaa *geeneihin*, jotka ovat ratkaisun osia eli muuttujia. Optimointitehtävässä etsitään geeneille hyviä arvoja.

Geenien mahdolliset arvot ovat *alleeleja*. Esim. binäärikromosomin geenit ovat bittejä, joiden alleelit ovat 1 ja 0. Jos jonkin muuttujan koodaamiseen on käytetty useampaa kuin yhtä bittiä, niin tämä vierekkäisten bittien jono muodostaa geenin. Geenillä on 2^n alleelia, missä n on geenin muodostamien bittien lukumäärä. Geenin paikkaa kromosomissa ilmaisee sen *lokus*, jota käytetään järjestysnumeron tapaan siten, että kromosomin ensimmäisen geenin lokus on 1.

Genotyyppi tarkoittaa yksilön geneettistä koostumusta, joka muodostaa sen ominaisuudet. Yleensä yksilö koostuu vain yhdestä kromosomista, jolloin yksilö on *haploidinen*. Tässä tapauksessa käytetään usein termejä 'kromosomi' ja 'yksilö' samassa merkityksessä. Voidaan käyttää myös ohjelmointitapaa, jossa yksilöllä on kaksi (*diploidinen*) tai useampia kromosomeja. *Genomi* tarkoittaa yksilön kaikkien geenien ja siten myös kaikkien kromosomien joukkoa.

2.1.2 Soveltaminen

Geneettisiä algoritmeja on usein sovellettu epälineaarisen optimoinnin tehtäviin silloin, kun ei ole tarpeen löytää globaalia optimia, vaan voidaan määritellä "riittävän hyvä" ratkaisu. Kuhunkin tehtävään sopivan matemaattisen mallin, tietorakenteen ja hyvyysfunktion muodostaminen on kuitenkin hankalaa. Ei ole yksiselitteistä tapaa hyödyntää geneettisiä algoritmeja mahdollisimman tehokkaasti, sillä käsiteltävän ongelman laatu voi vaikuttaa ratkaisumenetelmän tehokkuuteen huomattavasti.

Geneettisiä algoritmeja voidaan soveltaa mihin tahansa ongelmaan, jonka ratkaisujen käypyys ja hyvyys voidaan määritellä. Käypien ratkaisujen joukko eli jokainen mahdollinen populaation jäsen on voitava esittää yhdenmukaisessa muodossa, jotta uusia yksilöitä voidaan muodostaa selkeiden sääntöjen avulla. Populaation jäsenet asetetaan paremmuusjärjestykseen hyvyysfunktion avulla.

2.1.3 Ongelmia

Lokaalit optimit saattavat aiheuttaa ongelmia käytettäessä geneettisiä algoritmeja. Mm. populaation suuri koko ja yksilöiden homogeenisuus eli geneettinen samankaltaisuus saattavat hidastaa pääsyä pois lokaalista optimista. Tämä ongelma voidaan ratkaista lisäämällä geneettistä diversiteettiä eli monimuotoisuutta, mikä voidaan toteuttaa esimerkiksi käyttämällä useita rinnakkaisia populaatioita. (Simula 1997)

Vaikka geneettisiä algoritmeja on tutkittu paljon, on tärkeitä kysymyksiä vielä avoimina. Geneettisten algoritmien toimintaa on vaikea analysoida matemaattisesti ja usein niiden soveltaminen vaatii hyvien arvojen ja mekanismien etsimistä yrityksen ja erehdyksen kautta. Pyrkimyksenä on kopioida evoluution käyttämiä tehokkaita prosesseja, mutta luonnon adaptiiviset järjestelmät ovat monimutkaisia, eikä niitä vielä ymmärretä kovin hyvin. Useita luonnossa havaittuja mekanismeja ei vielä ole kokeiltu geneettisissä algoritmeissa lainkaan, tai kokeilujen tulokset eivät ole olleet rohkaisevia.

2.2 ALGORITMIN TOIMINTA JA RAKENNE

Luonnon evoluutio näyttäisi perustuvan ns. rakennuspalikkahypoteesiin (*Building Block Hypothesis*). Tämän hypoteesin mukaan hyvien ratkaisujen osia satunnaisesti yhdistelemällä saadaan huomattavalla todennäköisyydellä vielä parempia ratkaisuja (Alander 1998, 18). Geneettiset algoritmit pyrkivät hyödyntämään tätä olettamusta.

Yksinkertaisen geneettisen algoritmin toimintaperiaate on Hollandin (1975) mukaan seuraava:

1. **Aloitus**: Luo satunnainen joukko yksilöitä eli aloituspopulaatio.
2. **Valinta**: Karsi huonoimmat yksilöt. Jos riittävän hyvä ratkaisu on löytynyt tai optimointiin käytettävä aika on loppunut, lopeta.
3. **Yhdistely**: Yhdistele ratkaisuja satunnaisesti risteyttämällä ja lisää tarvittaessa satunnaisuutta mutaatiolla.

4. **Toisto**: Palaa askeleeseen 2.

Lopetuskriteeri, joka voi olla esimerkiksi ratkaisun tavoitearvo tai optimointiin käytettävä aika, määritetään tapauskohtaisesti.

2.2.1 Ohjelmointi

Geneettisten algoritmien ydin on yksinkertainen ja mahdollista toteuttaa millä tahansa ohjelmointikielellä muutamalla kymmenellä tai sadalla ohjelmarivillä (Alander 1998, 19). Sen sijaan hyvyysfunktion ohjelmointi voi olla huomattavasti suuritöisempää ja onkin usein ohjelmoinnin aikaa vievin osuus. Myös kuhunkin tehtävään sopivan tietorakenteen ja ohjelmointitavan löytäminen voi olla hankalaa.

Kromosomit ohjelmoidaan kokonais- tai reaalilukujonoiksi. Yleensä pyritään mahdollisimman yksinkertaiseen ohjelmointitapaan, mutta tehtävän luonteesta riippuen on joskus perusteltua käyttää monimutkaisiakin rakenteita. Useimmissa sovelluksissa käytetään binäärikromosomeja, mutta binääriohjelmoinnissa käytetyt periaatteet soveltuvat sellaisinaan myös kokonaisluku- tai jopa reaalilukuohjelmointiin. (Floudas & Pardalos 1998, 262-263) Binärilukujen käyttäminen on usein suositeltavaa, koska kaikki kokonaisluvut voidaan joka tapauksessa ohjelmoida bittivektoreina.

2.2.2 Yksilö

Yksilöä merkitään kromosomilla C, jonka pituus on N_C. Kromosomi voidaan purkaa N muuttujaksi x_i, $i = 1,...,N$, jotka ovat optimointisovelluksen luonnolliset muuttujat eli geenit. Binääriohjelmoinnissa jokaista muuttujaa x_i koodataan n_i:llä bitillä siten, että bittien kokonaislukumäärä on kromosomin pituus: $\sum n_i = N_C$. Todellisen muuttujan ohjelmointiin käytettyjen bittien määrä n_i on laskennallisista syistä tärkeä tekijä. (Floudas & Pardalos 1998, 263) Ohjelmointivaiheessa määritetään myös tehtävässä käytettävät parametrit, kuten mutaatiotodennäköisyys p_m ja risteytymistodennäköisyys p_c.

2.2.3 Populaatio

Yksilöt muodostavat populaation. Yksi populaatio edustaa yhtä sukupolvea, jota voidaan nimittää myös iteraatioaskeleeksi, koska uusi sukupolvi korvaa aina edellisen. Optimointi aloitetaan aloituspopulaatiosta, jonka generoimisessa on suotavaa käyttää hyväksi kaikkea ongelmaan liittyvää tietämystä, jotta aikaa ei myöhemmin tuhlaantuisi pelkästään huonojen vaihtoehtojen karsintaan. Erityistä huomiota kiinnitetään myös populaation diversiteettiin; kaikki optimiratkaisuun tarvittavat rakenneosat tulisi olla mukana aloituspopulaatiossa. Näin ollen yksilön tietorakenne vaikuttaa aloituspopulaation suuruuteen.

Jos ensimmäistä populaatiota luotaessa ei ole käytössä mitään erityistietämystä, käytetään satunnaista aloituspopulaatiota. Tällöin valitaan niin suuri populaation koko, että kaikki ratkaisuun tarvittavat rakenneosat saadaan mukaan. Jatkuvien hyvyysfunktioiden tapauksessa populaatio voi olla sitä pienempi, mitä vähemmän hyvyysfunktiolla on lokaaleja optimikohtia (Alander 1998, 40).

Optimaalinen populaation koko määräytyy mm. ratkaistavan ongelman vaativuuden mukaan. Hyvä lähtökohta on 50 yksilöä (mt. 26). Vaikeissa ongelmissa on syytä käyttää hieman suurempaa populaatiota riittävän diversiteetin takaamiseksi, mutta jos hyvyysfunktio on laakea ja sileä, riittää pienempikin populaatio. Jos hyvyysfunktion muoto ei ole tiedossa tai ongelmaan liittyy muita vaikeasti ennakoitavia piirteitä, käytetään usein varmuuden vuoksi hieman suurempaa populaatiota.

Suuren populaation käsitteleminen on hankalaa ja hidasta. Lisäksi poikkeavien yksilöiden kehittyminen saattaa hukkua massaan. Pienessä populaatiossa voi taas sattuman osuus kasvaa liian suureksi. Populaation kokoon liittyviä ongelmia voidaan välttää muodostamalla useita rinnakkaisia populaatioita, jolloin yksilöiden monipuolisuus lisääntyy, mutta hallittavana on yhden suuren populaation sijasta yksittäisiä pieniä populaatioita. Eri populaatioissa voidaan käyttää hieman erilaisia ge-

neettisiä algoritmeja tai niiden parametreja. Geneettistä perimää voidaan sitten sekoittaa siirtämällä satunnaisesti yksilöitä populaatioiden välillä. (Simula 1997)

Tutkittaessa geneettistä ohjelmointia (*Genetic Programming*) on havaittu, että jos rinnakkaisten populaatioiden välillä ei ole kommunikaatiota, ei rinnakkaisten populaatioiden käyttämisestä saada mitään lisähyötyä. Jos sen sijaan käytetään yksilöiden vaellusta eli niiden siirtymistä satunnaisesti rinnakkaispopulaatioiden välillä, ovat tulokset systemaattisesti parempia kuin käytettäessä vain yhtä populaatiota. Populaatioparametrien määrittely riippuu kyseessä olevasta ongelmasta ja ne määritellään erilaisiksi jokaiselle rinnakkaiselle populaatiolle. On myös havaittu, että yksilöiden vaellukseen käytettävät parametrit eivät riipu ongelman laadusta. (Fernández et al. 2002) Tällaisia parametreja ovat esim. populaatiosta toiseen siirtyvien yksilöiden määrä sekä iteraatioiden määrä vaellusten välillä. Vaeltamaan voidaan valita tietty osuus parhaista tai huonoimmista yksilöistä, tai valinnassa voidaan käyttää jotakin muuta periaatetta.

Optimointiin kuluva aika voidaan laskea kaavalla $N_{pop} \cdot N_{gen} \cdot T(f)$, missä N_{pop} on populaation koko, N_{gen} sukupolvien suurin sallittu määrä ja $T(f)$ aika, joka kuluu hyvyysfunktion arvioimiseen. Kaavasta havaitaan, että populaatioiden koon ja määrän välillä vallitsee täytyy tehdä kompromissi, koska jos molemmat ovat suuria, kasvaa optimointiin kuluva aika liian suureksi. Kulloinkin kyseessä olevasta ongelmasta riippuen valitaan joko suuret, monimuotoiset populaatiot, jotka tutkivat ratkaisuavaruutta laajasti, tai pienemmät populaatiot ja pidempi etsintäaika. Helpot tehtävät ratkeavat yleensä muutamassa sukupolvessa. Jos näyttää siltä, että sukupolvia tarvitaan paljon, saattaa käytetty populaation koko olla liian pieni (Alander 1998, 41).

2.2.4 Hyvyysfunktio

Hyvyys- eli kohdefunktion avulla määritetään populaation yksilöiden laatu. Kunkin kromosomin hyvyys riippuu sen kyvystä ratkaista kyseessä oleva ongelma. Geneettiset

algoritmit eivät aseta mitään rajoituksia kohdefunktiolle – se voi olla numeerinen funktio, laaja ohjelma (esim. jonkin laitteen rakenteen tai toiminnan simulaattori), testattava laite tai jopa ihmisen antama subjektinen arvio. Hyvyysfunktio on optimoinnin eniten laskentaa vaativa osa, sillä sen arvo voidaan joutua laskemaan tuhansia tai jopa miljoonia kertoja toivotun lopputuloksen saamiseksi. (Alander 1998, 19-21)

Matemaattinen hyvyysfunktio on muotoa $f(x_1,...,x_N)$. Sen avulla yksittäiset kromosomit asetetaan paremmuusjärjestykseen. Kromosomin hyvyys lasketaan purkamalla se ensin muotoon $(x_1,...,x_N)$, esim. bittivektorista reaaliluvuksi, ja sijoittamalla muuttujien arvot sitten hyvyysfunktion kaavaan. Hyvyysfunktiota skaalaamalla voidaan ylläpitää sopivaa kilpailua yksilöiden välillä koko algoritmin suorituksen ajan: Kun erot hyvyysarvoissa ovat liian suuret, skaalaus pienentää niitä. Jos taas hyvyysarvojen erot muuten olisivat liian pienet, voidaan skaalaamalla myös korostaa eroja parhaiden ja huonoimpien yksilöiden välillä. (Goldberg 1989, 122) Hyvyysfunktion skaalauksella voidaan vaikuttaa myös algoritmin konvergenssinopeuteen eli siihen, kuinka nopeasti populaatiosta tulee hyvin homogeeninen. Populaation homogeenisuus tarkoittaa sitä, että suurimmalla osalla yksilöistä on lähes samat arvot kaikilla muuttujilla.

Yksilöiden käypyyden huomioimisessa voidaan käyttää sakkofunktiota, joka rankaisee ei-käyvistä ratkaisun osista pienentämällä yksilön hyvyysarvoa. Rajoitetun optimoinnin tehtävissä sakkofunktion muodolla voi olla suuri vaikutus etsintätehoon. Saattaa olla tehokkaampaa sallia hieman epäkelpoja ratkaisuja kuin sakottaa ei-käypyydestä liikaa, sillä usein optimiratkaisu löytyy juuri käyvän alueen reunalta.

Geneettisten algoritmien on havaittu olevan raskaita ja hitaita, jos hyvyysfunktio on jatkuva ja suhteellisen tasainen. Tällaisissa tapauksissa on parempi käyttää yksinkertaisempia algoritmeja, jotka löytävät globaalin optimin nopeammin kuin geneettinen algoritmi. Sen sijaan hyvyysfunktion ollessa epäjatkuva tai sen sisältäessä

useita lokaaleja optimikohtia geneettiset algoritmit toimivat monia muita ratkaisualgoritmeja paremmin.

2.3 Operaattorit

Optimointiprosessi tapahtuu populaatiossa geneettisten operaattoreiden välityksellä. John Hollandin alkuperäinen geneettinen algoritmi käytti neljää operaattoria: valintaa, risteytystä, mutaatiota ja inversiota. Inversiossa eli kääntymisessä tietyt geenit vaihtavat paikkaa. Geenien lokus siis muuttuu, mutta tulkinnan tulee säilyä ennallaan, mikä aiheuttaa laskennallista hankaluutta. Inversion tarkoitus on saattaa toisiinsa toiminnallisesti liittyvät geenit yhteen, jotta ne todennäköisemmin selviäisivät risteytyksestä yhdessä. Tätä mekanismia käytetään nykyään vain harvoin, ja koska sen tuomia etuja ei ole voitu todentaa (esim. Mitchell 1996; Goldberg 1989), ei sitä käsitellä tässä tutkielmassa enempää.

Seuraavassa esitetään yleisimmin käytetyt geneettiset operaattorit lyhyesti (esim. Goldberg 1989, 10-14):

- **Valinta**: Yksittäisiä yksilöitä kopioidaan jollakin valintaperusteella suoraan seuraavaan sukupolveen.
- **Risteytys**: Seuraavaan sukupolveen kopioitujen yksilöiden ominaisuuksia yhdistellään.
- **Mutaatio**: Prosessiin lisätään satunnaisuutta, jotta valinnassa mahdollisesti kadotettuja ominaisuuksia voidaan saada takaisin populaatioon.

Valinta ajaa populaatiota konvergoitumaan hyvään ratkaisuun, kun taas risteytys ja mutaatio lisäävät populaation diversiteettiä pyrkien estämään ennenaikaisen konvergenssin lokaaliin optimiin. Konvergenssi on ennenaikainen, jos riittävästi etsintää ei ole tapahtunut ennen sitä. Eräs yleinen syy populaation ennenaikaiseen konvergoitumiseen on huono hyvyysfunktion skaalaus. (Floudas & Pardalos 1998, 263-264)

2.3.1 Valinta

Luonnon evoluutiossa valinta ilmenee siten, että hyvin ympäristöönsä sopeutuvat yksilöt saavat keskimäärin enemmän jälkeläisiä kuin huonosti sopeutuvat. Näin niiden ominaisuuksia ilmentävät alleelit yleistyvät populaatiossa, ja pitkän ajan kuluessa populaation geenipooli muuttuu siten, että se sopeuttaa yksilöitä paremmin vallitsevaan ympäristöön. Geneettisten algoritmien valintaoperaattori siirtää geneettistä materiaalia siten, että keskimäärin paremmat kromosomit tuottavat enemmän jälkeläisiä seuraavaan sukupolveen. Näin vaikutus on paljon nopeampi kuin luonnossa, ja jo muutamassa sukupolvessa populaation keskimääräinen hyvyysarvo voi kasvaa huomattavasti.

Geneettisten algoritmien periaatteiden mukaisesti koko sukupolvi väistyy seuraavan tieltä. Kaikille populaation yksilöille lasketaan hyötyfunktion avulla hyvyysarvot, joiden perusteella yksilöt voidaan asettaa paremmuusjärjestykseen. Yksilöitä valitaan seuraavan sukupolven muodostamiseen todennäköisyydellä, joka on verrannollinen niiden hyvyyteen. Näin ollen keskimääräistä paremmat yksilöt tulevat valituiksi suuremmalla todennäköisyydellä kuin huonommat. Valinnassa käytettyjä operaattoreita ovat mm. elitismi, rulettivalinta, kaksintaistelumalli ja kromosomipankki.

Yleisin puhtaissa geneettisissä algoritmeissa käytetty menetelmä on *rulettivalinta* (*Roulette Wheel Sampling*), jossa kunkin yksilön osuus rulettipyörästä on yksilön hyvyysarvo suhteessa kaikkien yksilöiden hyvyysarvojen summaan. Jokaisen yksilön todennäköisyys tulla valituksi seuraavaan sukupolveen riippuu sen hyvyysarvosta, koska rulettipyörä pysähtyy sitä todennäköisemmin tietyn yksilön kattamalle sektorille, mitä suurempi tämä alue on, eli mitä suurempi on kyseisen yksilön hyvyysarvo.

Uutta sukupolvea luotaessa rulettipyörää pyöräytetään N_{pop} kertaa, missä N_{pop} on populaation koko. Kun yksilö valitaan seuraavaan sukupolveen, siitä tehdään kopio pooliin odottamaan muiden geneettisten operaattoreiden toimintaa. Rulettivalinnassa paremmat yksilöt saavat odotusarvoisesti enemmän kopioita seuraavaan sukupolveen, mutta huonoimmillakin yksilöillä on positiivinen, joskin pieni todennäköisyys tulla

valituiksi. Tämän menetelmän huono puoli on se, että valinnan suorittamiseksi täytyy laskea populaation jokaisen yksilön hyvyys. Kuten luvussa 2.2.4 todettiin, on hyvyysfunktion laskeminen yleensä optimoinnin aikaavievin tapahtuma, joten rulettivalinnan käyttäminen hidastaa ongelmanratkaisua etenkin suurten populaatioiden tapauksessa.

Elitismiksi kutsutaan menettelytapaa, jossa uuteen sukupolveen valitaan edellisestä sukupolvesta parhaat yksilöt. Elitismi on puhtaasti heuristinen lisäys geneettisiin algoritmeihin, ts. sillä ei ole mitään tekemistä evoluution periaatteiden kanssa. Elitismi varmistaa, että jo löydettyjä hyviä ratkaisuja ei kadoteta ennen kuin parempia löytyy. Menetelmää käytetään usein yhdessä muiden valintamenetelmien kuten rulettivalinnan kanssa. Elitismiä käytettäessä populaation hyvyys voi ainakin aluksi parantua nopeammin kuin rulettivalinnassa (Simula 1997). Tämä metodi toimii parhaiten, jos optimoitava funktio on laakea ja mahdollisesti vain yksihuippuinen. Monen huipun tapauksessa elitismi saattaa johtaa populaation diversiteetin häviämiseen ja siten ennenaikaiseen konvergenssiin lokaaliin optimiin. Tämä voidaan välttää käyttämällä esim. *erilaisuuskriteeriä*: jos yksilö poikkeaa riittävästi muista populaation jäsenistä, se hyväksytään mukaan seuraavaan sukupolveen, vaikka sen hyvyysarvo ei olisikaan riittävän suuri.

Jos optimointiin kuluva aika on kriittinen resurssi, on valinnassa parempi käyttää rulettivalinnan sijasta esim. *"kaksintaistelumallia"*, jossa populaatiosta valitaan satunnaisesti kaksi tai useampia yksilöitä kerrallaan. Niiden hyvyysarvot lasketaan ja paras hyväksytään seuraavaan sukupolveen. Tämä on laskennallisesti tehokas menetelmä, koska kaikkien yksilöiden hyvyysarvoa ei välttämättä tarvitse laskea. Elitismiä ei voida yhdistää kaksintaistelumalliin menettämättä laskenta-ajan säästöä.

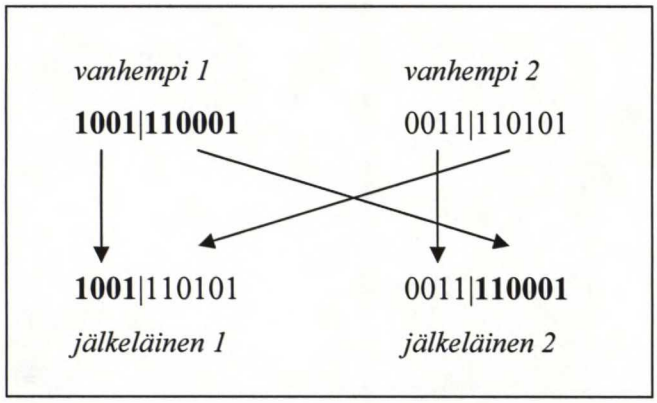
Sellaisissa ongelmissa, joissa yksilöt muodostuvat useammasta kuin yhdestä kromosomista, voidaan käyttää *kromosomipankkia*. Tällöin jonkin huonon yksilön erityisen hyvä genomin osa eli hyvin ratkaistu osaongelma voidaan ottaa talteen, vaikka yksilö ei siirrykään seuraavaan sukupolveen. Pankkiin varastoituja hyviä kappaleita

voidaan satunnaisesti yhdistää toisiin yksilöihin. (mt.) Jos yksilön kromosomit eivät edusta itsenäisiä osaratkaisuja, on kromosomipankin toiminnan tuloksia vaikea ennakoida.

2.3.2 Risteytys

Risteytys eli *rekombinaatio* on tyypillinen geneettisten algoritmien ominaisuus, joka erottaa ne muista stokastisista globaalin etsinnän menetelmistä. Risteytysoperaattori jäljittelee luonnossa tapahtuvaa haploidisten eli yksigeenisten organismien risteytymistä, jossa kahden kromosomin geneettinen informaatio yhdistetään vaihtamalla niiden osia keskenään. Risteytettävät yksilöt, vanhemmat, valitaan populaatiosta satunnaisesti luvussa 2.3.1 kuvattuja valintamenetelmiä käyttäen. Risteytyksessä saadut jälkeläiset voidaan joko siirtää suoraan seuraavaan sukupolveen, tai niiden hyvyys voidaan ensin arvioida ja valita jatkoon vain parempi jälkeläinen. Jälkeläisten hyvyyttä voidaan myös verrata vanhempien hyvyysarvoihin, jolloin seuraavaan sukupolveen siirtyvät vanhempien ja jälkeläisten joukosta parhaat yksilöt.

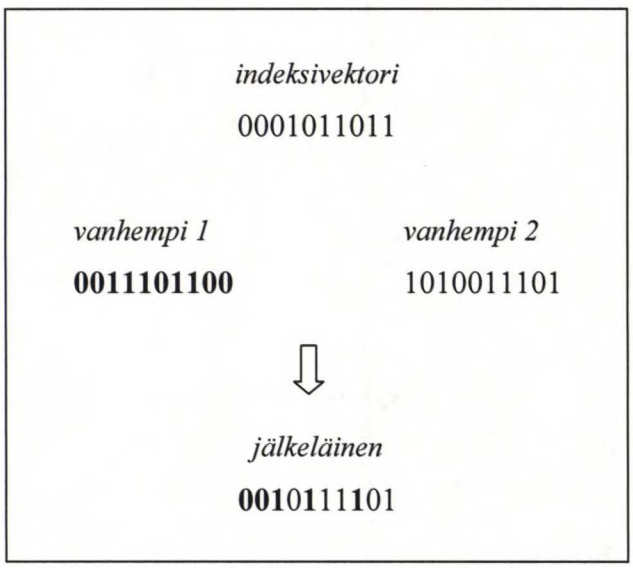
Yhden katkaisukohdan risteytyksessä valitaan vanhempien kromosomeille satunnainen katkaisukohta todennäköisyydellä p_c. Ensimmäiseen jälkeläiseen otetaan ensimmäisen vanhempikromosomin kaikki bitit katkaisukohtaan saakka ja toisen vanhempikromosomin kaikki bitit katkaisukohdan jälkeen. Toinen jälkeläinen saadaan yhdistämällä vanhempien jäljelle jääneet osat. Kuvassa 1 on esitetty yhden katkaisukohdan risteytys binäärikromosomeilla.



Kuva 1. Yhden katkaisukohdan risteytys binäärikromosomeilla

Yhden katkaisukohdan risteytys toimii hyvin, jos kromosomit ovat lyhyitä. Suurempien tietorakenteiden tapauksessa tämä menetelmä ei tuo tarpeeksi vaihtelua. Usein käytetäänkin *kahden katkaisukohdan risteytystä*, joka toimii samalla periaatteella kuin yhden katkaisukohdan risteytys, mutta tuo hieman enemmän satunnaisuutta ja kehittää populaatiota nopeammin. Voidaan käyttää myös useampia risteytymispisteitä, mutta yhden ja kahden katkaisukohdan menetelmät ovat vakiintuneita käytäntöjä.

Eräs esimerkki monipaikkaisesta risteytyksestä on useissa sovelluksissa tehokkaaksi havaittu *tasaristeytys* (*uniform crossover*). Tässä menetelmässä valitaan jälkeläisen jokainen alleeli satunnaisesti jommaltakummalta vanhemmalta. Käytännössä tasaristeytys toteutetaan esim. indeksivektorin avulla: indeksivektori on kromosomin pituinen, satunnaisesti generoitu bittivektori, joka kunkin geenin kohdalla ilmaisee, kummanko vanhemman alleeli kopioidaan jälkeläiseen. Tätä risteytysoperaattoria havainnollistaa kuva 2, missä indeksivektorin arvo 0 tarkoittaa, että ko. geenin alleeli otetaan ensimmäiseltä vanhemmalta ja arvo 1 sitä, että alleeli otetaan toiselta vanhemmalta.



Kuva 2. Tasaristeytys binäärikromosomeilla indeksivektorin avulla

Geneettiset algoritmit toimivat tehokkaimmin ainakin osittain separoituvien funktioiden optimoinnissa. Funktio on separoituva, jos $f(X) = f_1(x_1) + ... + f_n(x_n)$, eli funktio voidaan ilmaista usean yhden muuttujan funktion summana (Salo 1996, 159). Separoituvan funktion optimoinnissa voidaan päätyä tilaan, jossa jonkin yksilön hyvyysfunktion parametreista puolet on optimaalisia ja toinen puoli parametreista on optimaalinen jollakin toisella yksilöllä. Jos nämä yksilöt risteytyvät keskenään oikeasta kohdasta, yhden niiden jälkeläisen parametrien arvot muodostavat ongelman globaalin optimiratkaisun. (Floudas & Pardalos 1998, 264)

2.3.3 Mutaatio

Geneettinen informaatio on altis satunnaisille virheille eli mutaatioille. Luonnossa esiintyvät mutaatiot ovat yleensä haitallisia ja karsiutuvat geeniperimästä nopeasti. Geneettisten algoritmien operaattorina mutaation ensisijainen tehtävä on säilyttää populaation diversiteetti ja siten estää ennenaikainen konvergenssi lokaaliin optimiin. Mutaation tarkoitus on pystyä uudelleenluomaan populaatiosta valinnan ja risteytyksen kautta hävinneitä rakenneosia, jotta kaikki globaaliin optimiin tarvittavat osat pysyisivät optimointiprosessissa mukana. Ilman mutaatiota valinta karsii vähitellen pois kaikki

muut vaihtoehdot paitsi toistaiseksi parhaan löydetyn yksilön geenit, kun taas liian suuri mutaatioiden määrä hävittää löydettyjä elinkelpoisia kombinaatioita.

Mutaatio muuttaa satunnaisesti joidenkin geenien alleeleja. Binäärikromosomissa mutaatio ilmenee siten, että bitti muuttuu nollasta ykköseksi tai päinvastoin. Kuvassa 3 on esimerkki mutaatiosta binäärikromosomissa. Yleisimmässä mutaatio-operaattorissa mutaatiotodennäköisyys p_m on todennäköisyys, jolla jokainen yksittäinen bitti mutatoituu. Tämä todennäköisyys on jokaisella bitillä yhtä pieni, esim. $p_m = 0{,}001$.

alkuperäinen kromosomi	*kromosomi mutaation jälkeen*
0001101001	0001111001

Kuva 3. Lokuksen 6 mutaatio binäärikromosomissa

Jotta mutaatio-operaattorin toiminta olisi mielekästä, on se räätälöitävä tehtäväkohtaisesti. Kuten muistakin operaattoreista, on alan kirjallisuudessa esitetty myös mutaatiosta suuri määrä erilaisia variaatioita. Eräs mielenkiintoinen sovellus, joka havainnollistaa reaalilukujen käyttöä binäärikromosomien sijasta, on esitetty Siltasen pro gradu –tutkielmassa *Geneettisen algoritmin soveltaminen stokastiseen optimointiin* (1998). Siltasen käyttämässä mutaatio-operaattorissa suoritetaan mutatoitavaksi valitulle alleelille pienellä todennäköisyydellä radikaali mutaatio, jossa luku muuttuu komplementikseen. Muussa tapauksessa mutaatio aiheuttaa pienemmän muutoksen alleelin arvoon. Radikaali mutaatio aiheuttaa suuria muutoksia mutatoiduissa yksilöissä, mikä lisää populaation diversiteettiä tehokkaasti. Tämän menetelmän havaittiin kasvattavan algoritmin konvergenssinopeutta.

2.4 SKEEMATEOREEMA

Vaikka geneettisten algoritmien toimintaperiaate on yksinkertainen, tekee menetelmän epälineaarisuus ja diskreettisyys sen matemaattisesta analysoinnista vaikeaa (Alander

1998, 22). Tärkein teoreettisen tutkimuksen tulos on skeemateoreema (*the Schema Theorem*), joka selittää geneettisten algoritmien toimintaa matemaattisesti, eli esimerkiksi sitä, miksi algoritmi konvergoituu ja millaiset bittijonot säilyvät populaatiossa. Skeemateoreeman kehitti geneettisten algoritmien keksijä John Holland. Hän selitti geneettisten algoritmien toimintaa niiden kyvyllä prosessoida skeemoja eli samanlaisuuksia eri yksilöissä (Holland 1975).

Skeemat ovat jonoja, jotka muodostuvat merkeistä 1, 0 ja *, joka on ns. jokerimerkki (*don't care symbol*). Asteriski ilmaisee, että kyseisen lokuksen alleeli voi olla joko 0 tai 1. Goldberg (1989, 19) määrittelee asteriskin metasymboliksi eli symboliksi toisista symboleista: sitä ei koskaan käytetä eksplisiittisesti geneettisissä algoritmeissa, vaan se on vain keino havainnollistaa yhtäläisyyksiä kaikkien tietynpituisten ja tiettyä symbolimuotoa olevien kromosomien välillä. Esimerkiksi skeema 1***0 käsittää kaikki viiden bitin jonot, joiden ensimmäinen bitti on 1 ja viimeinen 0. Kyseisen skeeman *aste* (*order*) eli määriteltyjen bittien määrä on kaksi ja *määrittävä pituus* (*defining length*) eli kauimmaisten määriteltyjen bittien etäisyys on neljä. Yksilöiden kromosomit voidaan muodostaa skeemoista. Esimerkiksi jono 11100 sisältää mm. skeemat 111, *1100 ja 1*10. (esim. Mitchell 1996, 27-28)

Skeemateoreema esittää matemaattisesti skeemojen käyttäytymistä kromosomissa. Skeeman hyvyys on kaikkien sitä edustavien yksilöiden hyvyysarvojen keskiarvo. Skeemateoreema kertoo, että jos tietyn skeeman hyvyys keskimäärin kasvaa, sitä edustavien yksilöiden lukumäärä populaatiossa tulee kasvamaan eksponentiaalisesti. Tämä kasvunopeus riippuu kyseisen skeeman hyvyydestä suhteessa populaation keskimääräiseen hyvyyteen. (Floudas & Pardalos 1998, 264)

Seuraavassa esitetään skeemojen lisääntymisen ja vähenemisen dynamiikka populaatiossa Mitchellin (1996, 28-30) esimerkin mukaisesti. Olkoon H skeema, joka esiintyy populaatiossa ainakin kerran ajanhetkellä t. $m(H,t)$ kertoo skeeman H esiintymisten lukumäärän populaatiossa ajanhetkellä t ja $\hat{u}(H,t)$ on skeeman H

keskimääräinen hyvyys ajanhetkellä t. Silloin $E(m(H,t+1))$ on skeeman H esiintymisten lukumäärän odotusarvo populaatiossa ajanhetkellä $t+1$.

Rulettivalinnan mukaisesti kromosomin x jälkeläisten odotettu määrä on $f(x)/\bar{f}(t)$, missä $f(x)$ on kromosomin x hyvyys ja $\bar{f}(t)$ on populaation keskimääräinen hyvyys ajanhetkellä t. Jos oletetaan, että kromosomi x on populaatiossa hetkellä t ja siinä esiintyy skeema H, mutta jätetään huomioimatta risteytysten ja mutaatioiden vaikutus, saadaan määritelmän mukaisesti

$$E(m(H,t+1)) = \sum_{x \in H} f(x)/\bar{f}(t) = (\hat{u}(H,t)/\bar{f}(t))m(H,t),$$

koska $\hat{u}(H,t) = \left(\sum_{x \in H} f(x)\right) \Big/ m(H,t)$ kromosomille x ajanhetkellä t. Siten skeeman H esiintymisten lukumäärän kasvaminen tai väheneminen populaatiossa riippuu arvosta $\hat{u}(H,t)$ eli skeeman H keskimääräisestä hyvyydestä ajanhetkellä t.

Risteytykset voivat vähentää skeeman H esiintymisten lukumäärää populaatiossa. Olkoon p_c risteytymistodennäköisyys yhden katkaisukohdan risteytykselle ja oletetaan, että skeema H on valittu risteytykseen. Skeema H selviää risteytyksestä muuttumattomana, jos se esiintyy yhdessä jälkeläisistä. Tätä selviytymistodennäköisyyttä merkitään $S_c(H)$ ja sille saadaan alaraja kaavalla

$$S_c(H) \geq 1 - p_c\left(\frac{d(H)}{l-1}\right),$$

missä $d(H)$ on skeeman H määrittävä pituus ja l on ratkaisuavaruuden kromosomien pituus bitteinä. Risteytykset, jotka tapahtuvat skeeman H määrittävän pituuden sisällä eli kauimmaisten määriteltyjen bittien välissä, voivat tuhota skeeman eli tuottaa jälkeläisiä, joissa sitä ei esiinny. Siten siis kertomalla risteytymistodennäköisyydellä se osuus kromosomin bittijonosta, jonka skeema H käsittää, saadaan yläraja

todennäköisyydelle, jolla skeema tuhoutuu risteytyksessä. Kyseessä on yläraja siksi, että jotkut risteytykset skeeman määriteltyjen bittien sisällä eivät johda skeeman tuhoutumiseen (esim. kahden identtisen kromosomin risteytys). Kun tämä yläraja vähennetään 1:stä, saadaan alaraja skeeman H selviytymisen todennäköisyydelle. Kaavasta havaitaan, että risteytyksestä selviytymisen todennäköisyys on suurempi lyhyille skeemoille.

Myös mutaatiot voivat häiritä skeemaa H. Olkoon p_m todennäköisyys, jolla bitti mutatoituu ja $S_m(H)$ todennäköisyys, jolla skeema H selviää mutaatioista muuttumattomana. Tällöin

$$S_m(H) = (1 - p_m)^{o(H)},$$

missä $o(H)$ on skeeman H aste. Jokaiselle bitille todennäköisyys, jolla se ei mutatoidu, on $1 - p_m$. Siten siis todennäköisyys, jolla yksikään skeeman H määritellyistä biteistä ei mutatoidu, on yhden bitin todennäköisyys säilyä muuttumattomana korotettuna potenssiin, joka on skeeman määriteltyjen bittien määrä. Tästä kaavasta havaitaan, että mutaatioista selviävät muuttumattomina todennäköisemmin skeemat, joiden aste on pieni.

Skeemateoreemana tunnetaan kaava, joka saadaan ottamalla huomioon kaikki edellä mainitut skeeman H esiintymisten lukumäärään vaikuttavat tekijät:

$$E(m(H,t+1)) \geq \frac{\hat{u}(H,t)}{\bar{f}(t)} m(H,t)\left(1 - p_c \frac{d(H)}{l-1}\right)\left((1 - p_m)^{o(H)}\right).$$

Teoreemasta käy ilmi, että lyhyiden, alhaisen asteen skeemojen esiintymisten lukumäärä populaatiossa lisääntyy sukupolvesta toiseen likimain kaavan $\hat{u}(H,t)/\bar{f}(t)$ arvon verran, koska tällaiset skeemat muuttuvat risteytysten ja mutaatioiden seurauksena vain harvoin. Teoreema ilmaisee alarajan skeeman H odotetulle

lukumäärälle populaatiossa ajanhetkellä $t+1$, koska se ottaa huomioon vain risteytyksen ja mutaation haitalliset vaikutukset. Risteytystä pidetään kuitenkin voimakkaana geneettisten algoritmien ratkaisutehon lähteenä, koska sen kautta hyvät skeemat voivat yhdistyä toisiin hyviin, korkeamman asteen skeemoihin. Mutaatio puolestaan takaa diversiteetin säilymisen tietyssä lokuksessa. Skeemateoreema pätee skeemojen lisäksi mihin tahansa ratkaisuavaruuden kromosomien osiin. Yleensä keskitytään kuitenkin skeemoihin, koska ne kuvaavat hyvin niitä osaratkaisuja, joita yhdistelemällä saadaan luvussa 2.2 mainitun rakennuspalikkahypoteesin mukaisesti parempia ratkaisuja.

2.5 Arviointia ja sovellusaloja

Goldbergin (1989, 7-10) mukaan geneettiset algoritmit poikkeavat perinteisistä optimointimenetelmistä neljällä tavalla:

1. Itse muuttujien sijasta käytetään koodattuja muuttujasarjoja.
2. Optimia lähdetään etsimään usean yritteen populaatiosta eikä yksittäisestä ratkaisuavaruuden pisteestä. Pistepopulaation tutkiminen mahdollistaa useiden ratkaisuavaruuden huippujen tutkimisen rinnakkain, jolloin todennäköisyys juuttua lokaaliin optimiin pienenee.
3. Ratkaisuavaruudesta ei tarvita mitään muuta tietoa kuin hyvyysfunktio, jonka avulla ratkaisujen laatua arvioidaan. Etsinnän suuntaamiseen ei siis tarvita esimerkiksi derivaattoja tai gradientteja.
4. Geneettisten algoritmien käyttämät siirtymäsäännöt (*transition rules*) eivät ole deterministisiä vaan probabilistisia eli todennäköisyyteen perustuvia. Etsintä ei kuitenkaan ole satunnaishakua, vaan satunnaisuutta hyödynnetään etsinnän ohjaamisessa lupaavimmille ratkaisuavaruuden alueille.

Geneettisten algoritmien suorituskykyä on usein verrattu muihin heuristisiin menetelmiin. Yleensä ne ovat menestyneet vertailuissa hyvin, joskaan testit eivät aina ole olleet kovin kattavia. Tietotekniikan alalla ei myöskään ole ollut tapana julkaista negatiivisia tuloksia, sillä siitä ei voida päätellä paljoakaan, että menetelmällä ei ole saatu ennakoituja tuloksia – menetelmä voi olla huono, mutta yhtä hyvin jotakin muuta

on voinut mennä vikaan, mikä on ohjelmoinnissa melko yleistä. Sen sijaan hyvien tuloksien saaminen sattumalta tai viallisella ohjelmalla on epätodennäköistä, joten positiivisille tuloksille on perusteltua antaa paljon painoarvoa. (Alander 1998, 62)

Geneettisten algoritmien etu verrattuna muihin optimointimenetelmiin on niiden toimintaperiaatteen yksinkertaisuus. Lisäksi toimintaperiaate mahdollistaa laskennan hajauttamisen usealle prosessorille, mikä vähentää optimointiin kuluvaa aikaa. On havaittu, että geneettiset algoritmit löytävät hyviä ratkaisuvaihtoehtoja kaikille ongelmille, joskaan niiden tuottamien ratkaisujen mielekkyys tai optimaalisuus ei ole taattu. Usein ei myöskään pystytä sanomaan, kuinka lähellä globaalia optimia löytynyt ratkaisu sijaitsee (Reeves 1995). Tämä on kuitenkin tyypillistä kaikille heuristisille globaalin optimoinnin menetelmille.

Fogel (1999, 30-31) toteaa, että geneettiset algoritmit toimivat hyvin ongelmissa, joille ei tunneta hyviä ratkaisuja. Asiantuntijoita pyritään käyttämään ongelmanratkaisussa aina kun mahdollista, mutta eri asiantuntijat voivat olla eri mieltä tai vaihtaa mielipidettään, ja myös erehtyminen on inhimillistä. Geneettisillä algoritmeilla on potentiaalia pystyä ratkaisemaan ongelma, miten ratkaista ongelmia. Onkin herännyt kysymys, voitaisiko geneettisiä algoritmeja käyttää optimoimaan mm. toisten geneettisten algoritmien parametreja. Mitchell (1996, 158) huomauttaa, että geneettisen algoritmin koodaustavan lyöminen lukkoon ennen optimoinnin aloittamista on paradoksi: jos ongelma on niin vaikea, että siihen kannattaa käyttää geneettistä algoritmia, on mahdotonta tietää ongelmasta ennakkoon niin paljon, että voidaan löytää paras koodaustapa. Tämän vuoksi olisi tehokkaampaa käyttää geneettistä algoritmia sopeuttamaan koodaustapaa optimoinnin aikana.

Pulmallista geneettisissä algoritmeissa on niiden ominaisuuksien vajavainen tunteminen sekä suhteellisen pitkä suoritusaika, joka kuluu lähes täysin optimoitavan funktion laskentaan eikä itse optimointialgoritmin suorittamiseen. Tämä aika kuitenkin lyhenee jatkuvasti mikrotietokoneiden tehon parantuessa. Ehkä suurempi ongelma on se, että geneettiset algoritmit eivät ole yleispätevä, valmiita vastauksia antava menetelmä, vaan

niiden menestyksellinen käyttö vaatii yleistä optimointiajattelutavan hallintaa ja soveltamista kulloinkin kyseessä olevaan ongelmaan. Taulukkoon 1 on koottu geneettisten algoritmien tärkeimmät vahvuudet ja heikkoudet Alanderin mukaan (1998, 22).

Etuja	Haittoja
❖ robusti	❖ huonosti tunnettu
❖ yleinen	❖ hidas
❖ joustava	❖ vaatii optimointitietämystä
❖ yksinkertainen	❖ ei-analyyttinen
❖ ratkaisuja jatkuvasti saatavilla	❖ stokastinen

Taulukko 1. Geneettisten algoritmien tärkeimmät ominaisuudet
Lähde: mukaillen Alander 1998, 22

Geneettisiä algoritmeja voidaan käyttää kaikilla aloilla, joissa optimointi on tarpeen. Nykyään ei yleensä käytetä puhtaita geneettisiä algoritmeja, vaan niihin sovelletaan mahdollisimman paljon ratkaistavana olevaan ongelmaan tai kyseiseen alaan liittyvää erityisosaamista. Yksinkertaista geneettistä algoritmia ja siitä johdettuja variaatioita on käytetty menestyksekkäästi löytämään lähes optimaalisia ratkaisuja moniin teknisiin ja tieteellisiin sovelluksiin. Monimutkaisempia sovelluksia on kehitetty mm. molekyylibiologian, peliteorian, systeemiteorian sekä populaatiogenetiikan aloilla. Alander mainitsee Tekes-raportissaan (1998) erityisesti neuroverkot, jotka kuuluvat geneettisten algoritmien suosituimpiin sovellusalueisiin. Geneettisillä algoritmeilla on mm. optimoitu verkkojen rakennetta sekä opetettu verkkoja. Useissa sovelluksissa on myös käytetty erilaisia geneettisen algoritmin ja neuroverkon hybridejä. Taloustieteissä geneettisillä algoritmeilla on mallinnettu mm. innovaatioprosessia, tarjousstrategioita sekä markkinoiden muodostumista (Mitchell 1996, 16).

3 KULJETUSOPTIMOINTIONGELMA

Tässä tutkielmassa ratkaistaan geneettisellä algoritmilla kuljetusongelma, joka on yleistys Lallukan pro gradu –tutkielmassaan (2003) käsittelemästä raakapuun laivausongelmasta. Ongelmana on puunhankintayrityksen yhden vuoden aikana tapahtuvien aluskuljetusten optimointi. Tutkimuksessa käytettävä ongelmanasettelu noudattaa suurimmalta osaltaan Lallukan mallia ja oletuksia. Yrityskohtainen tieto on kuitenkin jätetty pois ja tilalle on otettu joitakin yleisiä oletuksia, koska tämän tutkimuksen ensisijainen tarkoitus ei ole tuottaa yksittäiselle yritykselle uutta strategista tietoa, vaan tutkia geneettisen algoritmin yleistä soveltuvuutta tämän tyyppisten ongelmien ratkaisuun.

3.1 Ongelman esittely

Optimoitavana ongelmana on puunhankintayrityksen kuljetukset Itämerellä siten, että hankintaverkoston tehtaiden kysyntä saadaan tyydytettyä annettujen rajoitteiden puitteissa mahdollisimman kustannustehokkaasti. Koska ongelmassa on sekä kysyntää että tarjontaa ainoastaan Itämeren satamakaupungeissa, on laivaus ainoa kuljetustapa, joka huomioidaan ongelman ratkaisussa; on realistista olettaa, että laivaus lyhyintä reittiä on kustannustehokkaampaa kuin muut kuljetusmuodot. Ongelman tärkeimmät elementit ovat alukset, satamat sekä kuljetettavat raakapuulajit (Lallukka 2003, 20). Seuraavissa kappaleissa esitellään nämä elementit lyhyesti mallin muodostamista varten.

Oletuksena on, ettei hankintayrityksellä ole omaa laivastoa. Kuljetuksiin käytettävä laivasto vuokrataan siis eri omistajilta rahtisopimuksilla. Markkinoilla on saatavilla useita eri tyyppisiä ja kokoisia aluksia, jotka voidaan jakaa tyyppeihin Stopfordin (1997, 383) näkemyksen mukaan: laivat, jotka ovat toistensa substituutteja, ovat samaa tyyppiä. Tässä ongelmassa huomioitavia aluksia on saatavilla 42 ja ne voidaan jakaa kolmeentoista tyyppiin (Lallukka 2003, 21-22). Merkitään laivojen tyyppejä symboleilla k1-k13 ja yksittäisiä aluksia kirjaimilla *a*, *b*, *c*... Koska alusten kokoa

rajoittavat satamissa monet asiat, kuten lastaukseen ja purkuun käytettävien nosturien toimintasäde sekä satamien mataluus, luokitellaan alustyypit suuriin, keskisuuriin ja pieniin aluksiin. Taulukossa 2 on lueteltu laivatyypit, alusten koodit, kuhunkin tyyppiin kuuluvien alusten määrät sekä koot.

Tyyppi	Alukset	Määrä	Koko
k1	k1a	1	keskisuuri
k2	k2a-c	3	keskisuuri
k3	k3a-e	5	pieni
k4	k4a-e	5	pieni
k5	k5a-e	5	pieni
k6	k6a-b	2	pieni
k7	k7a	1	suuri
k8	k8a-c	3	suuri
k9	k9a-c	3	pieni
k10	k10a-c	3	keskisuuri
k11	k11a-c	3	keskisuuri
k12	k12a-e	5	suuri
k13	k13a-c	3	pieni

Taulukko 2. Alustyypit ja koot

Lähde: mukaillen Lallukka 2003, 22

Tarjontasatamat sijaitsevat Venäjällä, Virossa, Latviassa ja Liettuassa yhteensä yhdeksässä satamassa. Kysyntää on Suomessa, Ruotsissa ja Norjassa yhteensä yhdeksässä satamassa. Kaikki mallissa käytettävät satamat eivät vastaa olemassaolevia satamia; joitakin pieniä satamia on yhdistetty tai niiden tarjontaa on siirretty suurempiin, lähellä sijaitseviin satamiin. Seuraavassa taulukossa on lueteltu tarjonta- ja kysyntäsatamat maittain. Lisäksi satamat näkyvät kartalla liitteessä 1.

Tarjontasatamat		Kysyntäsatamat	
Viro	Kunda	**Suomi**	Inkoo
	Pärnu		Kaskinen
	Tallinn		Rauma
Latvia	Liepaja		Sunila
	Mersrags	**Ruotsi**	Husum
	Riga		Piteå
Liettua	Klaipeda		Sundsvall
Venäjä	Pietari 1	**Norja**	Namsos
	Pietari 2		Oslo

Taulukko 3. Tarjonta- ja kysyntäsatamat

Kuljetettavana on kuutta erilaista puutavaraa: koivukuitua, havupuukuitua, haapakuitua, tukkeja, mekaanista haapaa sekä puuhaketta. Lallukan oletuksia noudattaen kukin alus kuljettaa vain yhtä puutavaralajia kerrallaan. Puuhake on kuitenkin poikkeus, sillä hakkeella lastatut alukset voivat kuljettaa kansilastinaan havupuukuitua (Lallukka 2003, 22). Jokaiseen kysyntäsatamaan viedään vain tiettyjä puutavaralajeja taulukon 4 mukaisesti.

SUOMI		RUOTSI		NORJA	
Satama	**Puulaadut**	**Satama**	**Puulaadut**	**Satama**	**Puulaadut**
Inkoo	Mekaaninen haapa	**Husum**	Koivukuitu	**Namsos**	Tukki
			Haapakuitu		
Kaskinen	Koivukuitu		Havupuukuitu	**Oslo**	Tukki
	Haapakuitu		Puuhake		
	Tukki				
		Piteå	Koivukuitu		
Rauma	Havupuukuitu		Havupuukuitu		
	Puuhake		Tukki		
	Tukki				
		Sundsvall	Koivukuitu		
Sunila	Havupuukuitu		Havupuukuitu		
	Puuhake		Puuhake		
	Tukki		Tukki		

Taulukko 4. Kysyntäsatamiin toimitettavat puulajit

3.2 Tavoitteet ja metodit

Ongelmassa on tavoitteena optimoida eri aluksilla kuljetettavien puutavaralajien määrä eri reiteillä yhden vuoden aikana. Optimaalinen ratkaisu tyydyttää kysynnän olemassaolevan tarjonnan ja annettujen toiminnallisten rajoitteiden puitteissa siten, että ratkaisun kokonaiskustannus minimoituu. Mallin muotoilulla ja rajoitteilla taataan, että ehdotettu ratkaisu on kustannustehokas ja toteuttaa asetetut rajoitteet.

Tulosten vertailun vuoksi tehtävässä GAMS-mallissa alusten kuljettamien puutavaralaatujen määrät ilmaistaan liukuluvuilla ohjelmiston toiminnan helpottamiseksi. Geneettisessä algoritmissa käytetään kuljetusmäärien kuvaamiseen kuitenkin kokonaislukuja (m^3). Koska ongelmassa on kyse suurista määristä – kymmenistä tai jopa sadoista tuhansista kuutioista – on kokonaislukuoletus hyvin perusteltu, eikä sen takia menetetä informaatiota. Koska C-ohjelmointikieltä eivät koske samantyyppiset rajoitukset kuin valmiita optimointiohjelmistoja, pyritään geneettisessä algoritmissa käyttämään mahdollisimman intuitiivisia yksiköitä ja realistisia oletuksia.

3.3 Aiheen rajaus ja rajoitukset

Kuten jo aiemmin on mainittu, tämän tutkimuksen tarkoituksena ei ole luoda strategista tietoa, vaan tutkia geneettisen algoritmin soveltuvuutta sellaisten ongelmien ratkaisuun, joissa on tarpeen huomioida monenlaisia reaalielämässä ilmeneviä rajoituksia. Tutkimuksen päätulos ei näin ollen ole puunlaivausongelman ratkaisu, vaan saadun ratkaisun laatu verrattuna GAMS:n antamaan tulokseen sekä itse optimointiprosessi.

Koska tutkimuksessa käytetty ongelma on vain esimerkki monista yrityselämän komplekseista päätösongelmista, ei tässä tutkielmassa pohdita mallissa käytettyjen oletusten validiteettia. Joiltakin osin ongelmaa on jouduttu yksinkertaistamaan hieman, mikä on tyypillistä vaikeiden ongelmien optimoinnissa, koska mikään valmis optimointiohjelmisto ei pysty käsittelemään kaikkia reaalimaailman ilmiöitä.

Geneettisessä algoritmissa on kuitenkin pyritty ottamaan huomioon mahdollisimman monta todellista rajoitetta, jotta päästäisiin realistisempiin tuloksiin.

Muutamia mallia koskevia rajoituksia on syytä nostaa esille. Ensimmäinen huomioitava seikka on se, että malli antaa kokonaiskustannuksen laivanvarustajien näkökulmasta, eikä se siten ole vuosittainen vuokrakustannus puunhankintayritykselle. Lisäksi mallissa käytettävien hintojen arviointi on vaikeaa, eivätkä ne välttämättä vastaa tämänhetkisiä todellisia kustannuksia. Toiseksi, mallia ei ole tarkoitettu operatiivisen toiminnan suunnitteluun lyhyellä aikavälillä, koska se antaa laivojen optimaalisen allokoinnin yhden vuoden ajalle. Tulos on siis luonteeltaan strateginen. Kolmanneksi, malli ei optimoi todellista logistiikkaoperaatiota vaan vain osaa siitä, sillä esim. lastaus-, purku- tai varastointikustannuksia ei oteta mallissa huomioon. (Lallukka 2003, 17-19) Nämä rajoitukset ovat toissijaisia tämän tutkimuksen tavoitteen kannalta, mutta on syytä pitää mielessä, että tämän tyyppisiin asioihin on kiinnitettävä huomiota todellisissa päätöstilanteissa.

Jos geneettisellä algoritmilla saadaan puunkuljetusongelmassa hyviä tuloksia kohtuullisessa ajassa, menetelmän voidaan olettaa toimivan hyvin monissa muissakin saman tyyppisissä ongelmissa. Näin ollen tämän tutkimuksen tulokset hyödyttävät ehkä enemmän esim. logistiikkayhtiöitä yleensä kuin yksittäistä hankintayritystä. Tulokset antavat viitteen siitä, ovatko geneettiset algoritmit hyvä ratkaisukeino komplekseihin logistiikkaongelmiin ja muihin tiukasti rajoitetuihin ongelmiin.

4 OPTIMOINTIMALLI

Luvussa 3 esitellystä puunkuljetusongelmasta muodostetaan tässä luvussa matemaattinen, diskreetin optimoinnin malli. Malli käsittää kaikki ongelman kannalta keskeiset tekijät. Tavoitteena on löytää mahdollisimman kustannustehokas allokointi käytettävissä olevalle laivastolle siten, että kysyntä tyydytetään olemassaolevan tarjonnan ja toiminnallisten rajoitteiden puitteissa. Tuloksena saadaan optimaaliset rahtimäärät kutakin puutavaralajia tarjontasatamista kysyntäsatamiin kullakin aluksella vuoden aikana.

Alusten allokointiin vaikuttavia tekijöitä ovat mm. kuljetettavat puutavaralajit ja niiden kysyntä- ja tarjontamäärät, reittien pituudet sekä kysyntä- ja tarjontasatamien infrastruktuuri. Malli noudattaa suurimmaksi osaksi Lallukan (2003, 23-25) muodostamaa mallia, mutta joitakin rajoitteita on jätetty pois, jotta ongelma on saatu yleisempään muotoon. Lisäksi malliin on sisällytetty joitakin realistisuutta lisääviä vaatimuksia. Seuraavissa kappaleissa esitetään mallin elementit, rajoitukset sekä matemaattinen formulointi.

4.1 Mallin elementit

Optimointimallin tärkeimmät elementit - alukset, satamat ja puutavaralajit - on esitelty luvussa 3.1. Näiden lisäksi olennaisia tekijöitä ovat kuljetuksista aiheutuvat kustannukset, laivojen kapasiteetit sekä satamien kysynnät ja tarjonnat. Näihin tekijöihin liittyvät oletukset esitetään tässä luvussa.

Raakapuun kuljetuksista aiheutuu sekä kiinteitä että muuttuvia kustannuksia. Alusten kiinteät kustannukset ovat päivittäisiä operointikustannuksia, jotka aiheutuvat aluksen ylläpidosta (Stopford 1997, 160). Tällaisia kustannuksia ovat esimerkiksi korjaus-, vakuutus- ja miehistökustannukset. Tässä tutkimuksessa käytetyt alusten kiinteät kustannukset on listattu liitteessä 2 alusten muiden teknisten tietojen kanssa.

Muuttuvat kustannukset ovat sidoksissa aluksen kulkemaan matkaan. Stopfordin (1997, 166) mukaan muuttuvia kustannuksia ovat mm. polttoaine-, satama-, hinaus- ja luotsauskustannukset. Tässä tutkimuksessa huomioidaan normaalista käytännöstä poiketen myös vakuutusmaksut muuttuvina kustannuksina, koska ne on määritelty matkakohtaisesti jokaiselle alukselle. Vakuutuskustannusten ei siis oleteta riippuvan esim. matkan pituudesta tai aluksen lastista, vaan summa on kiinteä ja peritään jokaista aluksella tehtyä matkaa kohden.

Alusten polttoaineenkulutus satamassa ja merellä on annettu liitteessä 2 tuhansina litroina päivässä. Alusten oletetaan käyttävän kahta polttoainelaatua: MGO:ta ja IFO 180:a. Suurin osa aluksista käyttää MGO:ta sekä merellä että satamassa, mutta alustyypit k7 ja k11 käyttävät IFO 180:a merellä ja MGO:ta satamissa. MGO:n hinnaksi on arvioitu 240 €/t ja IFO:n 160 €/t. Polttoaineiden muiden ominaisuuksien ymmärtäminen ei ole tämän tutkimuksen kannalta tärkeää, joten niihin ei tässä puututa.

Edellä mainittujen kustannusten lisäksi on huomioitava kaikki satamissa syntyvät kustannukset. Esimerkiksi Suomessa alukset maksavat jokaisesta satamakäynnistä sisään- ja ulosluotsausmaksun, satamamaksun, kiinnitys- ja irrotusmaksun sekä jätehuoltomaksut. Suomen, Ruotsin ja Norjan satamien kustannustiedot kullekin alustyypille on listattu liitteessä 3. Liitteessä 4 annetaan Venäjän ja Baltian maiden satamamaksut. Muiden satamakustannusten lisäksi Suomessa ja Ruotsissa on käytössä ns. väylämaksujärjestelmä (*fairway due system*), jonka mukaan alus joutuu maksamaan väylämaksun ensimmäiseltä 10 satamakäynniltä Suomessa ja ensimmäiseltä 12 satamakäynniltä Ruotsissa, minkä jälkeen maksua ei peritä. Järjestelmän vuoksi alusten ei ole järkevää seilata sekä Suomeen että Ruotsiin saman vuoden aikana, mikä huomioidaan mallin rajoitteissa (ks. luku 4.2 rajoite 11). Norjalla ei ole käytössä väylämaksujärjestelmää, joten jokainen alus voi seilata Norjaan riippumatta siitä, seilaako se myös Suomeen tai Ruotsiin.

Optimoinnissa käytetyt kysyntä- ja tarjontaluvut ovat todellista dataa, jota on satunnaisesti häiritty ± 25%. Alkuperäisen tilanteen mukaisesti kaikki kysyntä voidaan

tyydyttää mallissa huomioitavalla tarjonnalla. Kysyntä on annettu liitteessä 5 ja tarjonta liitteessä 6 satamittain ja puulaaduittain.

Alusten lastaus- ja purkuajat riippuvat kunkin aluksen kapasiteetista. Kapasiteetti luonnollisesti myös määrittelee, kuinka monta matkaa on tehtävä, jotta tietty määrä puuta saadaan kuljetettua satamasta toiseen. Aluksen kapasiteetti kuutiometreinä ei ole sama kaikille raakapuulajeille, koska paino kuutiometriä kohden ei ole vakio. Liitteessä 2 on ilmoitettu laivojen kapasiteetit kuutiometreinä kutakin puulajia.

Matka-ajat ja -kustannukset lasketaan mallissa edestakaista matkaa kohden, eli laivan oletetaan palaavan lastaussatamaan tyhjänä. Edestakainen matka-aika koostuu sekä merellä että satamassa vietetystä ajasta. Seilausaika saadaan kertomalla satamien välinen etäisyys (km, taulukko liitteessä 7) laivan nopeudella (km/h, liite 2) ja jakamalla se 24:llä, jolloin saadaan seilausaika vuorokausina. Laivojen nopeuden ei oleteta riippuvan lastin määrästä tai sääolosuhteista, vaan optimoinnissa käytetään alusten keskimääräistä nopeutta. Alusten lastaus- ja purkunopeuden oletetaan olevan 130 m^3/h jokaisessa satamassa. Tämä luku on keskimääräistä lastausnopeutta selvästi alhaisempi, mutta tällä tavalla pystytään jättämään varaa odottamattomille viivästyksille satamissa sekä muille satunnaistekijöille. Tietyntyyppiset alukset ovat rakenteensa ja teknisten ominaisuuksiensa takia nopeampia lastata ja purkaa, joten tyyppien k1, k2, k7, k10 ja k12 lastaus- ja purkunopeuden oletetaan olevan hieman suurempi, 150 m^3/h. Lisäksi lastausaikaa on kasvatettu vaikeiden talviolosuhteiden vuoksi Pärnussa 12 tunnilla ja Pietarissa 60 tunnilla.

4.2 Rajoitukset

Tehtävän rajoitukset voidaan jakaa kahteen ryhmään: loogiset ja operatiiviset rajoitteet (Lallukka 2003, 33-36). Loogiset rajoitteet takaavat kysynnän tyydyttämisen käytössä olevan tarjonnan ja kapasiteetin rajoissa. Loogisella rajoitteella asetetaan myös päätösmuuttujat eli kuljetettavien puulaatujen määrät epänegatiivisiksi kokonailuvuiksi. Operatiiviset rajoitteet liittyvät aluksille asetettuihin rajoituksiin esim. satamien

infrastruktuurin, varastointimahdollisuuksien tai raakapuun laadun vuoksi. Loogiset rajoitteet pätevät kaikissa kuljetusongelmissa, mutta operatiiviset rajoitteet riippuvat sovelluksesta. Ne täytyy asettaa tehtäväkohtaisesti, ja niissä käytettyjen oletusten validiteetti vaikuttaa tulosten realistisuuteen ja käytettävyyteen. Tässä luvussa selitetään ongelman rajoitteet, jotka esitetään matemaattisessa muodossa luvussa 4.3 samaa numerointia noudattaen.

Loogiset rajoitteet:

(1) *Kysyntärajoite* takaa, että kaikki kysyntä tyydytetään. Tämä rajoite asetetaan kaikille kysyntäsatamille ja raakapuulajeille.

(2) *Tarjontarajoite* takaa, että tarjontaa ei ylitetä. Rajoite asetetaan kaikille tarjontasatamille ja raakapuulajeille.

(3) *Kapasiteettirajoite* takaa, ettei alusten käyttöaikaa vuodessa ylitetä. Rajoitteessa lasketaan, kuinka monta vuorokautta kukin alus on käytössä vuoden aikana. Käyttövuorokausien määrä saadaan, kun kerrotaan edestakainen matka-aika matkojen määrällä. Matka-aika saadaan lisäämällä seilausaikaan purku- ja lastausajat. Seilausaika saadaan jakamalla satamien etäisyys aluksen nopeudella ja muuntamalla osamäärä vuorokausiksi. Matkojen määrä puolestaan saadaan jakamalla kuljetettava raakapuun määrä aluksen kapasiteetilla. Tämä luku pyöristetään ylöspäin seuraavaan kokonaislukuun, koska osittaisia matkoja ei ole mahdollista tehdä. Aluksen käyttöaika eli matka-ajan ja matkojen määrän tulo ei saa ylittää käytössä olevien vuorokausien määrää, joka on kerrottu binäärimuuttujalla. Tämä binäärimuuttuja saa arvon yksi jos ja vain jos kyseinen alus on mukana ratkaisussa. Kunkin aluksen oletetaan olevan käytössä 341 päivää vuodessa. Tämä oletus seuraa Stopfordin (1997, 179) laskelmista, joiden mukaan rahtilaivat ovat poissa käytöstä keskimäärin 24 päivää vuodessa mm. huoltotöiden ja lomien vuoksi.

(4) *Epänegatiivisuusrajoite* takaa, etteivät päätösmuuttujat eli kuljetettavien puulaatujen määrät saa negatiivisia arvoja. Lisäksi vaatimuksena on, että määrät ovat kokonaislukuja (m^3).

Toiminnalliset rajoitteet:

(5) *Alusten kokorajoite* estää tiettyjen alusten pääsyn liian mataliin tai kapeisiin satamiin. Kokorajoitteen saattavat aiheuttaa myös satamien tekniset ongelmat, jotka estävät tietyntyyppisten alusten lastaamisen. Kolmeen satamaan pääsy on rajoitettu: Mersragsiin, Pietari 2:een ja Pärnuun. Taulukossa 5 on esitetty alustyypit, jotka eivät voi seilata näihin satamiin.

Satama	Alustyypit
Mersrags	k2, k3, k7, k8, k11, k12
Pietari 2	k2, k3, k7, k8, k11, k12
Pärnu	k3, k11

Taulukko 5. Alustyypit, jotka eivät voi seilata Pärnuun, Mersragsiin tai Pietari 2:een

(6) *Inkoon mekaaninen haapa*: Inkooseen vietävällä mekaanisella haavalla on tiukat tuoreusvaatimukset. Tämän vuoksi vähintään 70% Inkooseen menevästä mekaanisesta haavasta täytyy kuljettaa pienillä aluksilla.

(7) *Norjan tukit*: Namsokseen ja Osloon vietäviä tukkeja ei voida varastoida pitkiä aikoja laatuongelmien vuoksi. Tästä syystä vähintään 80% Norjan satamiin menevistä tukeista on kuljetettava pienillä aluksilla.

(8) *Sunilan puuhake*: Suuret puuhake-erät aiheuttavat varastointiongelmia Sunilassa. Tämän vuoksi vähintään 80% hakkeesta on kuljetettava pienillä aluksilla, ja suurilla aluksilla ei voida kuljettaa haketta Sunilaan lainkaan.

(9) *Husumin puuhake*: Myös Husumin hakevarasto on pieni, joten enintään 30% Husumin puuhakkeesta voidaan kuljettaa suurilla aluksilla.

(10) *Kansilastirajoite* sitoo puuhake- ja kansilastit yhteen. Kuten luvussa 3.1 mainittiin, alukset, jotka kuljettavat puuhaketta, voivat viedä havupuukuitua kansilastinaan. Kansilastin maksimimäärän vuoden aikana on oltava suhteessa ruumalastin määrään: jos aluksen ruumalasti kattaa esim. 50% kapasiteetista, voidaan kannella kuljettaa vain 50% kansilastin kapasiteetista. Sallittu kansilastin määrä saadaan jakamalla kansikapasiteetti aluksen hakekapasiteetilla ja kertomalla osamäärä kuljetettavan puuhakkeen määrällä. Tämä rajoite seuraa alusten vakausvaatimuksesta, eli kansilasti ei saa olla liian painava suhteessa ruumalastiin. Kansilasti voidaan jättää myös vajaaksi tai tyhjäksi. Näin ei kuitenkaan todennäköisesti käy sen takia, että optimointimalli minimoi käytettävien alusten ja niillä tehtävien matkojen määrän, jolloin alukset seilaavat aina mahdollisimman täysinä.

(11) *Väylämaksurajoite* takaa, että sama alus ei seilaa sekä Suomeen että Ruotsiin. Mitä useammin laiva käy joko Suomen tai Ruotsin satamissa, sitä halvemmaksi tulee yhden matkan kiinteä kustannus. Tästä syystä kunkin laivan pääsy sekä Suomeen että Ruotsiin estetään kahden binäärimuuttujan avulla. Ensimmäinen binäärimuuttuja ilmaisee, seilaako laiva Suomeen, ja toinen, seilaako laiva Ruotsiin. Binäärimuuttujien summan on oltava korkeintaan yksi.

4.3 MATEMAATTINEN MALLI

Tässä luvussa esitetään puunkuljetusongelman optimointimalli matemaattisin notaatioin. Malli allokoi aluksia reiteille siten, että kokonaiskustannus minimoituu ja asetetut rajoitteet toteutuvat. Minimoitava kokonaiskustannus saadaan laskemalla yhteen käytössä olevien alusten vuosittaiset kiinteät kustannukset ja kuljetuksista aiheutuvat muuttuvat kustannukset. Tuloksesta nähdään, montako kunkintyyppistä

alusta tarvitaan kuljettamaan kysytty määrä raakapuuta vuoden aikana, sekä kuinka paljon mitäkin puulaatua aluksilla kuljetetaan reiteittäin. Mallin elementit on selitetty edellä luvussa 4.1 ja rajoitukset luvussa 4.2.

Muuttujat:

z_k = binäärimuuttuja, joka kertoo, onko alus k mukana ratkaisussa

x_{ijkp} = aluksella k kuljetettava puutavaralajin p määrä satamasta i satamaan j, kokonaisluku (m^3)

fin_k = binäärimuuttuja, joka kertoo, seilaako alus k Suomeen

swe_k = binäärimuuttuja, joka kertoo, seilaako alus k Ruotsiin

Vakiot:

i = tarjontasatama, $i = 1, ..., 9$

j = kysyntäsatama, $j = 1, ..., 9$

k = alus, $k = 1, ..., 42$

p = puutavaralaji, $p = 1, ..., 6$

f_k = aluksen k kiinteät kustannukset (€/vuosi)

c_{ijkp} = muuttuvat kustannukset alukselle k, joka kuljettaa puutavaralajia p satamasta i satamaan j (€/matka)

d_{jp} = puutavaralajin p kysyntä satamassa j (m^3)

s_{ip} = puutavaralajin p tarjonta satamassa i (m^3)

t_{ijk} = aluksen k edestakainen matka-aika reitillä ij (vrk)

cap_{kp} = aluksen k kuljetuskapasiteetti puutavaralajia p (m^3)

k_s = pienet alukset ($k_s \subset k$)

k_l = suuret alukset ($k_l \subset k$)

Min $\sum_k f_k z_k + \sum_i \sum_j \sum_k \sum_p c_{ijkp} x_{ijkp}$

ehdoin

(1) $\sum_i \sum_k x_{ijkp} \geq d_{jp}$ $\forall j, p$

(2) $\sum_j \sum_k x_{ijkp} \leq s_{ip}$ $\forall i, p$

(3) $\sum_i \sum_j \sum_p t_{ijk} \frac{x_{ijkp}}{cap_{kp}} \leq 341 z_k$ $\forall k$, missä

$\frac{x_{ijkp}}{cap_{kp}}$ pyöristetään ylöspäin seuraavaan kokonaislukuun

(4) $x_{ijkp} \geq 0$, kokonaisluku

(5) $x_{ijkp} = 0$, jos alus k ei sovellu reitille ij (ks. taulukko 5, luku 4.2)

(6) $\sum_i \sum_{k \in k_s} x_{i,Inkoo,k,p} \geq 0{,}7 \sum_i \sum_k x_{i,Inkoo,k,p}$ $\forall p$

(7) a) $\sum_i \sum_{k \in k_s} x_{i,Namsos,k,p} \geq 0{,}8 \sum_i \sum_k x_{i,Namsos,k,p}$ $\forall p$

b) $\sum_i \sum_{k \in k_s} x_{i,Oslo,k,p} \geq 0{,}8 \sum_i \sum_k x_{i,Oslo,k,p}$ $\forall p$

(8) a) $\sum_i \sum_{k \in k_s} x_{i,Sunila,k,hake} \geq 0{,}8 \sum_i \sum_k x_{i,Sunila,k,hake}$

b) $\sum_i \sum_{k \in k_l} x_{i,Sunila,k,hake} = 0$

(9) $\sum_i \sum_{k \in k_l} x_{i,Husum,k,hake} \leq 0{,}3 \sum_i \sum_k x_{i,Husum,k,hake}$

(10) $x_{i,j,k,kansi} \leq cap_{k,kansi} \frac{x_{i,j,k,hake}}{cap_{k,hake}}$ $\forall i, j, k$, missä

$\frac{x_{i,j,k,hake}}{cap_{k,hake}}$ pyöristetään ylöspäin seuraavaan kokonaislukuun

(11) $fin_k + swe_k \leq 1$ $\forall k$

5 GENEETTINEN ALGORITMI KULJETUSONGELMAAN

Geneettisiä algoritmeja voidaan kirjoittaa lähes millä tahansa optimointikielellä, rakenteellisella tai oliopohjaisella, joten kulloinkin käytettävä ohjelmointikieli riippuu tehtävän ja ohjelmointikielen ominaisuuksista sekä ohjelmoijan preferensseistä. Koska tässä tutkielmassa ratkaistava ongelma on monimutkainen ja vaatii paljon laskentaa, päätettiin geneettinen algoritmi kirjoittaa C-ohjelmointikielellä sen laskennallisen tehokkuuden vuoksi. C-kieltä käyttämällä säästetään optimointiprosessissa todennäköisesti huomattavan paljon aikaa.

Tässä luvussa käydään läpi geneettisen algoritmin luomista prosessina sekä selitetään sen toimintaa loogisella tasolla. C-kielistä lähdekoodia ei ole liitetty tutkielmaan ohjelman laajuuden vuoksi. Seuraavissa alaluvuissa käydään läpi algoritmin olennaisimmat elementit – yksilö, populaatio, hyvyysfunktio, operaattorit ja parametrit.

5.1 Yksilö

Yksilön tietorakenteessa täytyy ottaa huomioon seuraavat seikat: tarjonta- ja kysyntäsatamat, kuljetettavat puutavaralajit sekä käytössä olevat alukset. Näin ollen yksilön genotyypistä tulee lähtökohtaisesti neliulotteinen matriisi, jossa on $9 * 9 * 6 * 42 = 20\,412$ geeniä. On selvää, että tämänkokoisen rakenteen käsittely on työlästä ja hidasta, joten rakennetta oli pyrittävä karsimaan kuitenkaan menettämättä informaatiota. Geenien määrää saatiin vähennettyä huomattavasti, koska kaikkiin kysyntäsatamiin ei viedä kaikkia raakapuulajeja. Tällaisia redundantteja yhdistelmiä on turha pitää tietorakenteessa mukana. Joitakin geenejä pystyttiin myös asettamaan pysyvästi nolliksi, koska kaikki alukset eivät voi seilata Pärnuun, Mersragsiin tai Pietari 2:een (ks. taulukko 5, luku 4.2). Nämä passiiviset geenit oli kuitenkin järkevä pitää mukana genotyypissä teknisistä syistä. Havaittiin myös, että tietorakenteessa voidaan käsitellä yksittäisten alusten sijasta alustyyppejä. Näillä korjauksilla saatiin geenien lukumääräksi 1577, mikä on edelleen hyvin suuri tietorakenne. Jos kaikki informaatio kuitenkin halutaan säilyttää, ei tietorakennetta voida pienentää enempää.

Liitteessä 8 esitetään osa yksilön genotyypistä. Yksilö koostuu yhdestätoista kromosomista: Ensimmäisessä kromosomissa on 13 geeniä, jotka määrittävät, kuinka monta kunkin tyyppistä alusta ratkaisussa on mukana. Arvot ovat kokonaislukuja nollan ja saatavilla olevan tyyppikohtaisen maksimimäärän väliltä (ks. taulukko 2, luku 3.1). Toinen kromosomi on saman tyyppinen kuin ensimmäinen, mutta se ilmaisee, moniko käytössä olevista aluksista seilaa Suomeen. Loput seilaavat Ruotsiin. Tällä tavalla huomioidaan mallin kaksi binäärimuuttujaa *fin* ja *swe*, jotka takaavat väylämaksurajoitteen toteutumisen. Yksilön loput yhdeksän kromosomia vastaavat kukin yhtä kysyntäsatamaa, ja niiden geenit kuvaavat kustakin tarjontasatamasta kuljetettavia raakapuulajeja kullakin alustyypillä. Kromosomien pituus vaihtelee sen mukaan, montaako puulajia kyseiseen satamaan kuljetetaan ja monestako satamasta kutakin puulajia saadaan. Pisimmän kromosomin, 403 geeniä, muodostaa Husum, johon viedään neljää raakapuulajia. Alusmääriä kuvaavien kahden ensimmäisen kromosomin jälkeen lyhyimmät kromosomit edustavat Namsoksen ja Oslon tukkikuljetuksia. Näissä kromosomeissa on vain 26 geeniä, koska tukkeja saadaan ainoastaan Klaipedasta ja Pietari 1:stä.

Yllä mainittujen kromosomien lisäksi yksilön tietorakenteessa on mm. tietokenttiä populaation numerolle ja yksilön hyvyysarvolle sekä kokonaislukuvektorit kansilastien huomioimista varten. Kansilastit päätettiin jättää optimoinnin ulkopuolelle ja olettaa, että aina kun ruumassa kuljetetaan puuhaketta, viedään kannella täysi sallittu määrä havupuukuitua. Tämä on alusten optimaalisen käytön kannalta perusteltu oletus. Kansilastin määrät lasketaan hyvyysfunktion evaluoinnin aikana ja niitä pienennetään vasta, jos tarjontaa ei ole riittävästi tai kysyntä ylitetään.

Geenien arvot olivat aluksi epänegatiivisia kokonaislukuja ilman mitään muita rajoituksia. Tämä kasvatti kuitenkin hakuavaruuden liian suureksi, eikä geneettinen algoritmi päätynyt arvoihin, jotka olisivat muodostaneet järkeviä kuljetusyksiköitä. Tämän vuoksi määritettiin päätösmuuttujille lisärajoitteita siten, että geenien arvot ovat aina täysiä laivalasteja. Tämä johti ylivientiin, joten myöhemmin lisättiin rajoitteita

siten, että geenien arvoiksi sallitaan vajaita laivalasteja, jos kysyntä tulee tyydytettyä tai tarjonta loppuu.

Kunkin tyyppisten alusten käytössä oleva lukumäärä oli aluksi yksi optimointiin osallistuvista kromosomeista, kuten myös kromosomi, joka määrittää, mitkä alukset seilaavat Suomeen. Tämä havaittiin huonoksi ratkaisuksi, sillä alusmäärät eivät muodostaneet yhteensopivia kokonaisuuksia kuljetusmäärien kanssa. Alusmääräkromosomit voidaan jättää optimoinnin ulkopuolelle, koska geenien arvot eli kuljetusmäärät voivat suoraan määrittää, montako tietyn tyyppistä alusta kuljetuksiin tarvitaan ja moniko niistä seilaa Suomeen.

On huomioitava, että käytetystä tietorakenteesta johtuen samassa lokuksessa olevien geenien tulkinta vaihtelee kromosomeittain. Esimerkiksi kolmannen kromosomin ensimmäinen geeni kuvaa k1-tyyppisillä aluksilla Pärnusta Kaskisiin kuljetettavaa koivukuidun määrää, kun taas kymmenennen kromosomin ensimmäinen geeni kuvaa k1-tyyppisillä aluksilla Klaipedasta Namsokseen kuljetettavaa tukkien määrää. Tällaiset eroavuudet kromosomeissa johtavat koodin määrän kasvuun, mutta samalla tietorakenne on huomattavasti pienempi ja tietokoneen muistia tarvitaan vähemmän ohjelman suoritukseen.

Yksilön tietorakenne oli vaikea kehittää, koska siinä on huomioitava lähes kaikki mallin elementit. Jokaisen geenin arvo vaikuttaa kysyntä-, tarjonta- ja kapasiteettirajoitteisiin, minkä lisäksi täytyy erikseen huomioida luvussa 4.2 selitetyt rajoitteet 6-9, jotka asettavat ehtoja käytettävien alusten koolle. Näin ollen ongelma ei ole separoituva; siitä ei ole erotettavissa itsenäisiä kokonaisuuksia, joita geneettinen algoritmi voisi optimoida ja yhdistellä menestyksekkäästi. Tällainen asetelma ei ole geneettisten algoritmien toimintaperiaatteiden kannalta kovin lupaava.

5.2 Populaatio

Yksilön tietorakenteen luomisen jälkeen haastavinta oli mielekkään aloituspopulaation luominen. Aloituspopulaation tulisi olla mahdollisimman laadukas, jotta saadaan parempia ratkaisuja nopeammin. Tässä vaiheessa onkin syytä käyttää hyväksi kaikkea, mikä ongelmasta tiedetään etukäteen.

Geneettisen algoritmin ensimmäinen versio aloituspopulaation luomisessa ei ollut menestyksekäs. Geenien arvoiksi generoitiin täysin satunnaisia kokonaislukuja nollan ja maksimikysynnän tai -tarjonnan väliltä. Jotta tällä tavalla olisi saatu aloituspopulaatioon kaikki optimiratkaisuun tarvittavat rakenneosat, olisi populaation koko pitänyt kasvattaa järjettömän suureksi. Näin ollen aloituspopulaation luomisprosessiin lisättiin useita rajoituksia järkevien lukujen saamiseksi aloitusyksilöihin.

Geenien arvoille asetetiin rajoite, jonka mukaan arvojen tulee olla kokonaisia laivalasteja. Jotta ylimääräinen vienti saataisiin estettyä, sallittiin arvoiksi vajaita laivalasteja sellaisissa tapauksissa, joissa geeniin liittyvien satamien kysyntä tai tarjonta ylittyy. Tällaisissa tapauksissa valitaan geenin arvoksi kysynnästä ja tarjonnasta pienempi. Arvojen generoinnin myötä pidetään kirjaa jäljellä olevasta tyydyttämättömästä kysynnästä sekä käytetystä tarjonnasta kussakin satamassa. Kun esimerkiksi jonkin sataman kaikki havupuukuitu on viety, asetetaan jäljellä olevien kyseisestä satamasta lähtevien havupuukuitukuljetusten arvoksi nolla. Samoin, jos esimerkiksi puuhakkeen kysyntä on jo tyydytetty, ei haketta enää viedä kyseiseen satamaan. Samaan tyyliin pidetään kirjaa eri alustyyppien jäljellä olevasta kapasiteetista.

Aluksi geenien arvot oli mahdollista generoida järjestyksessä, koska rajoituksista ei pidetty kirjaa. Kun aloituspopulaation luomisessa alettiin pitää kirjaa jo allokoiduista kuljetuksista ja asettaa kuljetusmääriksi nolla kysynnän tai tarjonnan täyttyessä, tuli tarpeelliseksi generoida geenien arvot täysin satunnaisessa järjestyksessä. Alusten käytön optimoimiseksi päätettiin lisätä vielä tekniikka, jossa aluksia käsitellään tyyppi

kerrallaan satunnaisessa järjestyksessä. Kullekin alustyypille arvotaan, montako kyseisen tyyppistä alusta ratkaisuun otetaan. Sen jälkeen näiden alusten kapasiteetti täytetään satunnaisesti ennen seuraavan alustyypin arpomista. Näin saadaan aloituspopulaatioon mahdollisimman tehokkaassa käytössä olevia aluksia. Tällä tekniikalla pyrittiin luomaan yksilöihin tehokkaita osaratkaisuja ja ideaa pyrittiin hyödyntämään myös geneettisissä operaattoreissa. Ei kuitenkaan tutkittu, syntyykö optimaalinen yksilö tällaisista osaratkaisuista.

Edellä kuvattujen rajoitteiden implementoinnin lisäksi aloituspopulaatioon luotiin ns. kuljetusoptimaalisia yksilöitä. Nämä yksilöt perustuvat kuljetusmääriin, jotka saadaan, kun optimoidaan ongelma GAMS:llä huomioiden pelkät kysyntä- ja tarjontarajoitteet sekä satamien väliset etäisyydet. Kaikki aluksia koskeva data ja rajoitteet jätetään siis huomioimatta. GAMS:n antamien lukujen perusteella muodostetiin testiajoissa aloituspopulaatioon kuljetusoptimaalisia yksilöitä tietty osuus populaation koosta. Kuljetusoptimaalisten yksilöiden avulla saadaan nopeammin käypiä ja suhteellisen hyviä ratkaisuja.

Tässä luvussa mainituilla rajoituksilla ja tekniikoilla pystytään luomaan aloituspopulaatioon lähes käypiä yksilöitä, jolloin lähtökohtainen asetelma on huomattavasti parempi kuin satunnaisen aloituspopulaation tapauksessa. Aloituspopulaation luomiseen tarvittavan koodin määrä tosin myös kasvaa – lopullisen ohjelman koodiriveistä n. 30% kuluu aloituspopulaation luomiseen. Koska aloituspopulaatio luodaan vain kerran, sen luomiseen kuluva aika ei kuitenkaan ole iso tekijä optimointiprosessissa.

Aloituspopulaation luomisen lisäksi populaation koko ja iteraatioiden määrä ovat tärkeitä tekijöitä. Nämä riippuvat käytettävän ongelman laadusta ja tyypistä. Koska puunkuljetusongelma on monimutkainen ja sillä on useita lokaaleja optimeita, on populaation koon oltava melko suuri ja iteraatioita tarvitaan useita satoja. Optimointiin kuluva aika luonnollisesti kasvaa populaation koon myötä, mutta suurempi populaatio johtaa hakuavaruuden laajempaan tutkimiseen, mikä puolestaan vähentää lokaaliin

optimiin juuttumisen todennäköisyyttä ja optimin löytämiseen tarvittavaa iteraatioiden määrää.

5.3 Hyvyysfunktio

Hyvyysfunktio on tässä tehtävässä, kuten tyypillistä geneettisille algoritmeille, optimoinnin eniten laskentaa vaativa osa. Hyvyysfunktion laskennallisesta massiivisuudesta kertoo se, että ohjelman koodiriveistä n. 40% kuluu populaation evaluointiin.

Optimointitehtävässä on monenlaisia kustannuksia ja rajoitteita, jotka on otettava viimeistään hyvyysfunktiossa huomioon. Hyvyysfunktiota käytetään määrittämään, edustaako yksilö käypää ratkaisua. Jos yksilö ei toteuta kaikkia rajoitteita, se saa sakkopisteitä. Sakot lasketaan eri tavoin riippuen rikotusta rajoitteesta. Yksilön hyvyysarvo saadaan yhdistämällä sakkopisteet yksilön edustaman ratkaisun kokonaiskustannukseen. Hyvyysarvon perusteella yksilöt voidaan asettaa paremmuusjärjestykseen.

Osa tehtävän rajoitteista on huomioitu jo yksilön genotyypissä. Hyvyysfunktiossa huomioitavia rajoitteita ovat satamien kysyntä- ja tarjontarajoitteet (ks. rajoitteet 1 ja 2, luku 4.2), laivojen kapasiteettirajoitteet (rajoite 3) sekä puun tuoreus- ja varastointivaatimuksista johtuvat alusten kokorajoitteet (rajoitteet 6-9). Kysyntä- ja tarjontarajoitteiden sekä alusten kokorajoitusten rikkomisesta seuraavat sakkopisteet riippuvat kuutiomäärästä, jolla rajoitetta rikotaan. Kapasiteettisakon suuruuteen taas vaikuttaa se, monellako vuorokaudella aluksen maksimikapasiteetti ylitetään.

Ongelmasta suoraan saatavien rajoitteiden lisäksi geneettisessä algoritmissa käytetään muutamia erikoisrajoituksia optimoinnin tehostamiseksi. Jotta alukset olisivat mahdollisimman tehokkaassa käytössä, lasketaan yksilön evaluoinnin yhteydessä kunkin alustyypin käyttöaste. Yksilöä sakotetaan sitä enemmän, mitä tehottomammin aluksia käytetään. Jos alus on käytössä yli 95%:sti, sakkokerrointa pienennetään. Jos

taas aluksen käyttöaste on alle 70%, sakkokerrointa kasvatetaan. Tämä rajoitus lisättiin, jotta geneettinen algoritmi pyrkisi allokoimaan vähemmän aluksia ratkaisuihin. Aluksi uskottiin, että alusten kiinteät kustannukset olisivat rajoittaneet alusmäärää, mutta testit osoittivat, että lisärajoite on tarpeen.

Hyvyysfunktiossa käytettävien sakkojen suuruuden asettamiseen ei ole olemassa sääntöjä tai ohjeita. On kokeiltava tehtäväkohtaisesti, millaisilla sakkojen arvoilla saadaan paras tulos. Tässä tutkimuksessa käytettiin adaptoituvia sakkokertoimia apuna sakkojen suuruuden määrittämisessä: kutakin sakkokerrointa kasvatettiin 10% jos yli 80% yksilöistä rikkoi rajoitetta ja pienennettiin 10% jos alle 5% yksilöistä rikkoi rajoitetta. Sakkoja pienentäminen tarvittaessa perustuu siihen, että optimiratkaisu löytyy tyypillisesti käyvän alueen reunalta. Tämän vuoksi on hyvä pitää hieman ei-käypiä yksilöitä populaatiossa. Adaptoituvat sakkokertoimet auttoivat sakkojen suuruusluokan määrittämisessä, mutta lopullisessa geneettisessä algoritmissa niistä ei havaittu olevan hyötyä.

5.4 Operaattorit ja parametrit

Geneettisessä algoritmissa käytetään operaattoreina valintaa, risteytystä ja mutaatiota. Koska tietorakenne on varsin suuri ja optimoitava tehtävä on monimutkainen, on operaattorit pyritty räätälöimään tehtävän rakenteeseen sopiviksi. Kustakin operaattorista kokeiltiin testiajoissa useita versioita, jotka esitellään luvuissa 5.4.1-5.4.3.

Vaikka geneettisiä algoritmeja on tutkittu paljon ja sovellettu monissa erilaisissa ongelmissa, ei ole olemassa yleispäteviä sääntöjä parametrien arvoista. Voidaan todeta, että mikään parametrien arvojen yhdistelmä ei toimi kaikissa ongelmissa, vaan hyvät arvot on etsittävä ongelmakohtaisesti. Yleensä on mahdotonta sanoa etukäteen, millaiset parametrien arvot toimivat kyseessä olevassa ongelmassa. Näin ollen puunkuljetusongelman optimoinnissa kokeillaan useita eri parametrien kombinaatioita, jotta saadaan paras mahdollinen tulos. Parametrikombinaatioita käsitellään luvussa 5.4.4.

5.4.1 Valinta

Valinta siirtää yksilöitä pooliin odottamaan risteytystä ja mutaatiota. Valintaoperaattoreina käytetään elitismiä ja rulettivalintaa. Eliittiä ei kopioida pooliin, vaan se siirretään suoraan seuraavaan sukupolveen. Elitismi saa testiajoissa arvot 5%, 10%, 20% ja 30%, ja loput seuraavan sukupolven muodostamiseen osallistuvista yksilöistä valitaan rulettivalinnalla.

Rulettivalinnassa yksilö tulee valituksi todennäköisyydellä, joka on suhteessa sen hyvyysarvoon. Koska kyseessä on minimointitehtävä, yksilöiden hyvyysarvot pitää ennen rulettivalintaa normalisoida siten, että paras yksilö saa suurimman normalisoidun hyvyysarvon, ja normalisoitujen hyvyysarvojen summa on yksi. Yksilön *j* normalisoitu hyvyysarvo lasketaan kaavalla:

$$norm\_fitness_j = \frac{\sum_i fitness_i - fitness_j}{(pop\_size - 1) * \sum_i fitness_i}.$$

Rulettivalinnassa arvotaan satunnaisluku väliltä [0,1[. Yksilöitä käydään läpi järjestyksessä paremmasta huonompaan ja samalla pidetään kirjaa niiden normalisoitujen hyvyysarvojen summasta. Se yksilö, jonka kohdalla summa ylittää arvotun satunnaisluvun, valitaan pooliin odottamaan risteytystä ja mutaatiota. Rulettivalinnasta kokeiltiin myös sellaista versiota, jossa valittavaksi tulee vain eliittiin kuuluvia yksilöitä. Tämä kuitenkin hävittää diversiteetin populaatiosta liian nopeasti, eikä menetelmällä saatu hyviä tuloksia.

5.4.2 Risteytys

Risteytys on geneettisten algoritmien tärkein operaattori, koska sen tehtävä on yhdistellä ratkaisukandidaattien hyviä rakenneosia. Puunlaivaustehtävä ei ole separoituva, eikä siitä ole muodostettavissa selkeitä kokonaisuuksia, joita olisi helppo yhdistellä mielekkäästi. Niinpä sopivan risteytysoperaattorin kehittäminen oli haasteellista.

Risteytysoperaattorista kokeiltiinkin useita erilaisia versioita ja niiden variaatioita. Risteytystekniikasta riippumatta risteytykseen otetaan kerrallaan kaksi yksilöä poolista siten, että jokainen poolissa oleva yksilö osallistuu risteytykseen tasan kerran. Tietty osuus pareista risteytetään ja loput yksilöt siirretään sellaisinaan seuraavaan sukupolveen.

Geneettisissä algoritmeissa yleisimmin käytetyt risteytysmenetelmät ovat yhden ja kahden katkaisukohdan risteytys sekä tasaristeytys. Kaikkia näitä menetelmiä kokeiltiin tässä tehtävässä. Yhden katkaisukohdan risteytyksessä alusmääriä kuvaavat kromosomit sekä jokainen yhdestätoista kysyntäsatamaa kuvaavasta kromosomista risteyttiin yhdestä satunnaisesti valitusta kohdasta. Tämä menetelmä ei toiminut kovin hyvin, koska se ei tuota riittävästi vaihtelua näin pitkien kromosomien tapauksessa. Kahden katkaisukohdan risteytys tuotti hieman parempia tuloksia, mutta ei pärjää vertailussa minkään kehittyneemmän menetelmän kanssa. Ongelma lienee sama kuin yhden katkaisukohdan tapauksessa: kromosomit ovat liian pitkiä, jotta tällainen menetelmä saisi riittävästi vaihtelua aikaan.

Huomattavasti parempia tuloksia saadaan tasaristeytyksellä, joka on toteutettu indeksivektorin avulla. Tästä risteytysmenetelmästä kokeiltiin perusversiota, jossa luodaan oma indeksivektori jokaiselle kromosomille. Näin saadaan parempia tuloksia kuin kahden katkaisukohdan risteytyksellä. Useimmat muut kokeillut risteytysmenetelmät ovat erilaisia versioita tasaristeytyksestä – niissä käytetään samaa indeksivektoria tietyille loogisille geeniryhmille.

Kysyntäperustaiseksi valintaristeytykseksi kutsutaan tässä menetelmää, jossa alusten määriä kuvaavat kromosomit risteytetään indeksivektorilla ja loput kromosomit kysyntälohkoittain sen mukaan, kumman vanhemman geenien summa on lähempänä vastaavaa kysyntälukua. Kysyntälohkolla tarkoitetaan kromosomin osaa, joka vastaa yhtä puutavaralajia. Esimerkiksi Kaskinen-kromosomin geenit 1-117 kuvaavat koivukuitukuljetuksia kaikista yhdeksästä tarjontasatamasta (ks. liite 8) ja muodostavat siten yhden kysyntälohkon. Valintaristeytys parantaa populaatiota nopeasti, mutta ei luo

uusia kombinaatioita kysyntälohkojen sisällä. Optimaaliseen ratkaisuun tarvittavat kysyntälohkot pitäisi siis olla mukana aloituspopulaatiossa ja mutaation pitäisi pystyä uudelleenluomaan hävinneitä lohkoja. Kysyntälohkot eivät kuitenkaan muodosta itsenäisiä kokonaisuuksia, joten menetelmän käyttö ei ole perusteltua.

Geneettisessä algoritmissa kokeiltiin myös kysyntäperustaisen valintaristeytyksen ja kahden katkaisukohdan risteytyksen tai tasaristeytyksen yhdistelmää siten, että jokainen kysyntälohko risteytettiin ja jälkeläiseen valittiin se lohko, joka paremmin tyydytti kysynnän. Näin saatiin uusia kombinaatioita ja pystyttiin tuottamaan nopeasti uusia, suhteellisen hyviä ratkaisuja. Jonkinlaisen valintamenetelmän käyttäminen risteytyksessä ei kuitenkaan ole luonnon periaatteiden mukaista. Tästä syystä kysyntäperustaisia valintaristeytysmenetelmiä ei hyväksytty lopullisiin testiajoihin.

Tasaristeytys alustyypeittäin noudattaa tasaristeytyksen perusversion periaatteita, mutta siinä arvotaan kolmentoista bitin indeksivektori, jonka bitit vastaavat alustyyppejä. Tämä menetelmä perustuu ideaan, jonka mukaan tehokkaasti käytetyt alukset muodostavat loogisia kokonaisuuksia geneettisen algoritmin yhdisteltäväksi. Alustyypeittäin tapahtuvasta tasaristeytyksestä kokeiltiin kahta erilaista menetelmää: saman indeksivektorin käyttämistä koko yksilölle ja oman indeksivektorin generoimista kullekin kromosomille. Normaalista tasaristeytyksestä poiketen luodaan kaksi jälkeläistä, joista parempi valitaan jatkoon. Jälkeläisten hyvyyttä voidaan myös verrata vanhempien hyvyyteen ja valita jatkoon näistä neljästä paras. Tämä menetelmä ei sanottavasti parantanut tuloksia, joten se ei ole käytössä algoritmin lopullisessa versiossa. Eri tasaristeytysmenetelmillä saadaan nopeasti käypiä ratkaisuja. Nämä ovat ainoat testiajoissa mukana olleet risteytysmenetelmät, koska muut kokeillut menetelmät havaittiin huonommiksi jo algoritmin kehitysvaiheessa.

5.4.3 Mutaatio

Aluksi geneettisessä algoritmissa käytettiin mutaatio-operaattoria, joka muuttaa jokaisen geenin arvoa satunnaisesti pienellä todennäköisyydellä. Koska yksilöllä on

suuri määrä geenejä, tämä johtaa siihen, että yksilön selviäminen mutaatiosta muuttumattomana on häviävän pieni. Tämän vuoksi muutettiin mutaatiota siten, että se valitsee tietyn osuuden eliittiin kuulumattomista yksilöistä mutatoitavaksi. Mutaatioon valittujen yksilöiden jokainen geeni mutatoidaan samalla, pienellä todennäköisyydellä. Mutatoitavan geenin uusi arvo on satunnainen määrä täysiä laivalastillisia. Pienin arvo on nolla ja suurin geeniä vastaava kysyntä tai tarjonta, riippuen siitä kumpi on pienempi. Jos uusi arvo on kysyntä tai tarjonta, sen ei tarvitse vastata täysiä laivalasteja.

Mutaatiosta kokeiltiin myös versiota, jossa geenin saama uusi arvo riippuu sen vanhasta arvosta. Jos vanha arvo on positiivinen, tulee uudeksi arvoksi nolla. Jos vanha arvo on nolla, uudeksi arvoksi tulee geeniä vastaavasta kysynnästä ja tarjonnasta pienempi. Samalla muutetaan muut vastaavan lohkon luvut nolliksi. Esim. jos mutatoitava geeni saa uudeksi arvokseen täyden kysynnän, muutetaan kaikki muut saman kysyntälohkon luvut nolliksi, jotta kyseiseen satamaan ei viedä liikaa. Jos taas tarjonta on pienempi kuin kysyntä, geeni saa uudeksi arvokseen tarjontaluvun, ja muut kyseistä tarjontasatamaa vastaavat 12 geeniä muutetaan nolliksi. Ongelmia tämän mutaatiotyypin kanssa tuottavat suurten lukujen aiheuttamat kapasiteetin ylitykset. Tällaiset mutaatiot ovat haitallisia. Joka tuontisatamaan voidaan kuitenkin teoriassa viedä kaikki yhdenlaatuinen puu yhdellä alustyypillä, joten tämän ei pitäisi tuottaa ylitsepääsemättömiä ongelmia. Menetelmä vaikuttaa intuitiivisesti järkevältä ja toimiikin testiajoissa huomattavan hyvin.

5.4.4 Parametrit

Parametrien arvojen vaihtelulla havaittiin olevan suuri vaikutus algoritmin toimintaan. Pienilläkin muutoksilla saatetaan joutua tilanteeseen, jossa käypää ratkaisua ei enää löydy tuhannessa iteraatiossa. Alander väittää, että geneettinen algoritmi ei ole herkkä parametrien vaihteluille (1998, 27). Tämä saattaa päteä usein käytetyille yksinkertaisille testiongelmille ja separoituville ongelmille, joiden rakenteeseen geneettiset algoritmit soveltuvat erittäin hyvin, mutta tässä tutkimuksessa havaittiin, että vahvasti

strukturoidussa ongelmassa parametrien arvojen asettaminen on työläs yritys-erehdys – prosessi.

Populaation koon annetaan vaihdella välillä 100-500 sadan yksilön välein ja aloituspopulaatioon muodostetaan 50%, 75% tai 100% kuljetusoptimaalisia yksilöitä. Erilaisia risteytyksiä ja mutaatioita on testiajoissa molempia kolme. Risteytystodennäköisyydelle p_c annetaan testiajoissa arvot 0,5; 0,75; 0,9 ja 0,95. Mutaatiotodennäköisyyteen p_m vaikuttaa kolme tekijää: elitismiaste, mikä osuus mutatoitavista yksilöistä mutatoidaan, sekä yksittäisen geenin mutaatiotodennäköisyys. Elitismille annettiin arvot 5%, 10%, 20% ja 30%. Eliitin ulkopuolisista yksilöistä mutatoitiin joko 30% tai 70%, ja yksittäisen geenin mutaatiotodennäköisyys oli 0,001; 0,01 tai 0,05. Kokonaismutaatiotodennäköisyys saa siten 4 * 2 * 3 = 24 eri arvoa, joista pienin on 0,00049 ja suurin 0,012. Mutaatiotodennäköisyys on hyvin pieni verrattuna alan muihin sovelluksiin. Mutaatiolla on kuitenkin tärkeä merkitys optimoinnissa: jos mutaatio jätetään algoritmista kokonaan pois, ei algoritmilla saada yhtä hyviä tuloksia kuin käyttämällä pientä mutaatioastetta.

Jos kaikkien edellä mainittujen operaattoreiden ja parametrien kombinaatiot halutaan testata, pitäisi ohjelma ajaa yhteensä 51 840 kertaa. Tämä on kuitenkin vain epärealistinen alaraja testiajojen määrälle, koska toimivat sakkokertoimet täytyy yleensä asettaa erikseen joka kombinaatiolle. Ei ole järkevää eikä kohtuullista käydä kaikkia kombinaatioita läpi kirjoittamatta tarkoitukseen soveltuvaa ajuria. Tässä tutkielmassa se ei kuitenkaan ole tarkoituksenmukaista, joten kombinaatioiden kokeilussa käytetään intuitiota ja aiemmin saatuja testituloksia. Seuraavassa luvussa esitellään geneettisellä algoritmilla saadut tulokset.

6 TULOKSET

Tässä luvussa käydään läpi geneettisellä algoritmilla saatuja tuloksia. Tärkeintä ei ole puunlaivausongelmaan löydetyn ratkaisun kokonaiskustannus, vaan se, miten hyvin geneettinen algoritmi suoriutui tehtävästä. Luvussa 6.1 tiivistetään eri parametrikombinaatioilla saatuja tuloksia ja luvussa 6.2 puhutaan populaation kehittymisestä optimoinnin aikana. Luku 6.3 vertaa geneettisellä algoritmilla saatua ratkaisua GAMS-ohjelman antamaan globaaliin optimiin. Tulosten merkitystä pohditaan luvussa 7.

6.1 PARAMETRIKOMBINAATIOT

Algoritmin lopullista versiota ajettiin useilla eri operaattoreiden ja parametrien arvojen kombinaatioilla, jotta löydettäisiin mahdollisimman tehokas yhdistelmä. Koska operaattoreiden toiminta on kytköksissä toisiinsa, on optimaalinen kombinaatio etsittävä kokeilemalla eri yhdistelmiä sen sijaan, että voitaisiin kokeilla jokaista operaattoria yksinään ja optimoida parametreja yksi kerrallaan. Algoritmi näyttää testiajojen perusteella olevan melko herkkä parametrien arvojen muutoksille. Myös sakkokertoimilla voidaan muuttaa optimoinnin kulkua selvästi. Kun parametrien arvoja tai operaattorikombinaatiota muutetaan, joudutaan sakkokertoimet yleensä asettamaan uudestaan käyvän ratkaisun löytämiseksi.

Taulukossa 6 esitetään joidenkin eri parametrikombinaatioiden vaikutus parhaan löydetyn yksilön kokonaiskustannukseen sekä optimointiin kuluvaan aikaan, kun optimointi lopetetaan 500 iteraation jälkeen. Tulokset on saatu 3GHz prosessorilla ja suoritusaika pitenee huomattavasti, jos laskentatehoa vähennetään. Taulukosta havaitaan, että vaihtelu parhaan löydetyn ratkaisun kokonaiskustannuksessa ei ole suuri; kaikki arvot ovat alle 30 miljoonaa euroa.

Risteytys-metodi	Mutaatio-metodi	Pop. koko	Kuljetusopt. yks. %	Elitismi %	Risteytys-tn %	Mutaatio-tn %	Kokonais-kust. (€)	Aika (s)
3	1	400	100	20	90	0,056	27 423 528	53
			75				29 427 870	52
			50				28 490 391	53
			100			0,24	29 578 801	48
						0,12	28 532 484	52
					95	0,056	28 177 041	54
					75		28 466 452	49
					50		28 073 716	44
				30	90	0,049	28 331 770	47
				10		0,063	29 941 014	59
				5		0,067	29 588 380	62
		500		20		0,056	28 452 975	67
		300					28 306 349	39
		200					28 151 248	26
		100					29 000 299	13
	2	400	100	20	90	0,056	29 639 069	56
	3						29 764 195	52
2	1	400	100	20	0,9	0,00056	28 489 208	51
	2						28 423 084	54
	3						29 500 465	51
1	1	400	100	20	0,9	0,00056	27 558 139	62
	2						28 853 379	66
	3						28 625 211	62

Taulukko 6. Parametrien vaikutus optimointiin 500 iteraatiossa

Taulukossa käytetyt risteytys- ja mutaationumerot tarkoittavat seuraavia metodeita:

Risteytys

1) Tasaristeytys, jossa arvotaan oma indeksivektori jokaiselle kromosomille. Risteytyksessä luodaan yksi jälkeläinen.
2) Risteytys alustyypeittäin. Arvotaan yksi 13 geenin indeksivektori, jonka perusteella yksilöiden kaikki kromosomit risteytetään 13 geenin lohkoissa. Risteytyksessä luodaan kaksi jälkeläistä, joista hyvyysarvoltaan parempi jatkaa seuraavaan sukupolveen.
3) Risteytys alustyypeittäin kuten edellä, mutta jokaiselle kromosomille arvotaan oma 13 geenin indeksivektori. Jälkeläisiä luodaan kaksi, joista parempi valitaan jatkoon.

Mutaatio

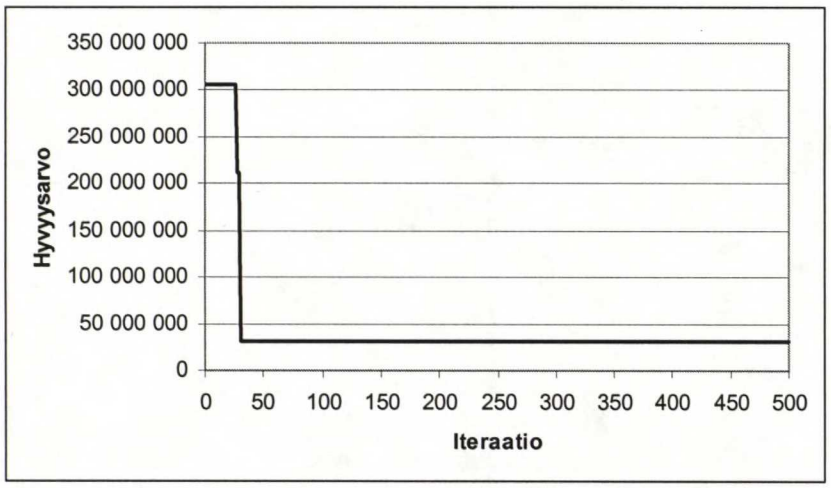
1) Mutatoitavaksi valittu geeni muutetaan lohkoa vastaavaksi kysynnäksi tai tarjonnaksi riippuen siitä, kumpi on pienempi. Samalla muut vastaavan kysyntä- tai tarjontalohkon geenit saavat arvon nolla.
2) Mutatoitavan geenin arvoksi muutetaan satunnainen luku annetulta väliltä siten, että arvo vastaa täysiä laivalasteja. Minimi on nolla ja maksimi on lohkoa vastaavasta kysynnästä ja tarjonnasta pienempi.
3) Mutatoitavan geenin uusi arvo on aina joko nolla tai annettu maksimi kuten edellä. Jos geenin vanha arvo on nolla, se saa annetun maksimiarvon. Muussa tapauksessa uusi arvo on nolla.

Nämä metodit on esitelty edellä luvuissa 5.4.2 ja 5.4.3.

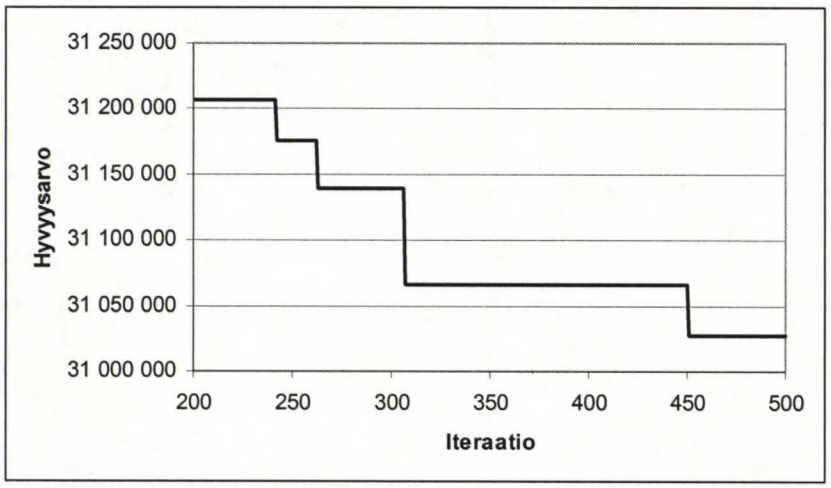
Paras löydetty kombinaatio geneettisen algoritmin operaattoreille on siis risteytysmetodi 3 ja mutaatiometodi 1 sekä 20%:n elitismi. Populaation koko on 400 ja aloituspopulaatio sisältää pelkkiä kuljetusoptimaalisia yksilöitä. Risteytystodennäköisyys on 0,9. Mutatoitavaksi valitaan eliittiin kuulumattomista yksilöistä 70%, joiden jokainen geeni mutatoidaan todennäköisyydellä 0,001. Näin ollen yksittäisen geenin kokonaismutaatiotodennäköisyys on 0,00056. Optimointi kestää vajaan minuutin.

6.2 Optimoinnin kulku

Seuraavissa kuvissa esitetään hyvyysarvon ja kokonaiskustannuksen kehittyminen parhaaseen ratkaisuun johtavan optimoinnin aikana. Kuvasta 4 nähdään, että hyvyysarvo paranee aluksi todella nopeasti: vajaan 50 iteraation aikana päästään jo todella lähelle lopullista hyvyysarvoa. Kuvassa 5 esitetään hyvyysarvon kehittyminen tarkemmin viimeisten 300 iteraation aikana. Kuten kuvista havaitaan, algoritmille on tyypillistä, että se jää pitkäksi ajaksi johonkin lokaaliin optimiin, kunnes jokin rekombinaatio tai mutaatio aiheuttaa etsinnän siirtymisen uuden lokaalin optimin läheisyyteen.

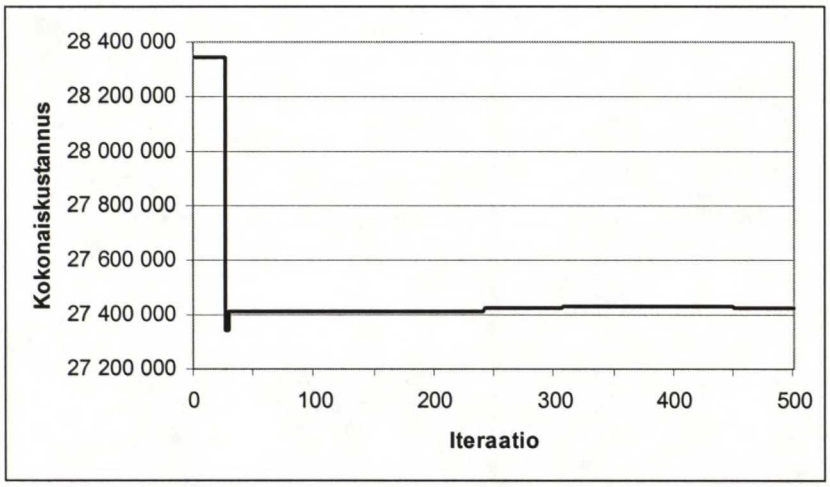


Kuva 4. Hyvyysarvon kehittyminen 500 iteraatiossa



Kuva 5. Hyvyysarvon kehittyminen iteraatioissa 200-500

Kuvasta 6 nähdään populaation parhaan yksilön kokonaiskustannuksen kehitys 500 iteraation aikana. Kustannus laskee aluksi rajusti ja käy alle 27,4 miljoonassa eurossa, mutta nousee sen jälkeen lähelle lopullista kokonaiskustannusta. Kustannus nousee optimoinnin aikana minimilukemasta, koska algoritmi hakee käypää ratkaisua. Kokonaiskustannus kehittyy hyvyysarvon tavoin hyppäyksittäin ja samassa arvossa pysytään pitkään.



Kuva 6. Parhaan yksilön kokonaiskustannuksen kehittyminen 500 iteraatiossa

6.3 VERTAILUA

Parhaan geneettisen algoritmin löytämän ratkaisun kokonaiskustannus on 27 423 528 € ja ratkaisu on esitetty liitteessä 9. Ratkaisussa on käytössä 13 alusta, joista kuusi seilaa Suomeen. Alusten käyttöaste on keskimäärin 96%. GAMS löytää tehtävän globaalin optimin ja antaa kokonaiskustannukseksi 23 943 918 € (liite 10). Tässä ratkaisussa aluksia on käytössä 11, joista viisi seilaa Suomeen. Geneettinen algoritmi näyttää siis juuttuvan lokaaliin optimiin, jonka kustannukset ovat 14,5% suuremmat kuin globaalin optimiratkaisun. Tosin GAMS:n antama kokonaiskustannus on vain alaraja todelliselle kustannukselle alla mainitusta syystä.

Eri menetelmillä saatujen tulosten vertailussa on huomioitava, että geneettinen algoritmi pystyy kuvaamaan ongelman realistisemmin kuin GAMS-malli. GAMS ei pysty huomioimaan aluksilla tehtävien matkojen kokonaislukuvaatimusta: matkojen määrä lasketaan jakamalla kuljetettava puumäärä aluksen kapasiteetilla, jolloin saadaan tuloksena liukulukuja. Geneettisessä algoritmissa tämä osamäärä pyöristetään ylöspäin seuraavaan kokonaislukuun, jolloin luvut vastaavat kokonaisia matkoja. Tämän tyyppisistä tekijöistä johtuen GAMS:n antama ratkaisu ei ole geneettisen algoritmin hyvyysfunktiolla arvioituna käypä: se rikkoo sekä kysyntä-, tarjonta- että

kapasiteettirajoitteita. Ratkaisujen kokonaiskustannusten eroa voidaan siten pitää vain ylärajana todelliselle erolle.

Geneettisen algoritmin antamassa ratkaisussa käytetään viittä samaa alusta kuin GAMS:n globaalissa optimissa. Molemmissa ratkaisuissa on käytössä neljä suurta alusta. Globaalissa optimiratkaisussa pieniä aluksia on vain kaksi, kun taas geneettisen algoritmin antamassa ratkaisussa niitä on viisi. Geneettinen algoritmi ei ilmeisesti pysty löytämään kustannustehokkaimpia alustyyppejä, koska globaalissa optimissa on käytössä kaikki tyypin 10 alukset, mutta geneettinen algoritmi ei päätynyt käyttämään näistä yhtäkään. Yllättävää on se, että vaikka kaikki aloituspopulaation yksilöt olivat kuljetusoptimaalisia, eivät puunkuljetusreitit parhaassa ratkaisussa ole kovin optimaalisia. Esimerkiksi Kaskisiin viedään koivukuitua mm. Klaipedasta ja Pietarista, kun taas globaalissa optimissa Kaskisten koivukuidun koko kysyntä pystytään tyydyttämään lähempänä sijaitsevien satamien tarjonnalla.

Geneettinen algoritmi on hyvin raskas, eikä sen suoritusaika pärjää vertailussa GAMS:n kanssa. Optimointiin kuluva aika ei kuitenkaan ole kohtuuton: 3GHz prosessorilla paras ratkaisu löytyy vajaassa minuutissa. Laskentatehon jatkuvasti kasvaessa ja halventuessa voidaan olettaa, että optimointiin kuluva aika ei ole suurin ongelma. Enemmänkin ongelmia tuottaa geneettisen algoritmin räätälöinnin työläys.

7 JOHTOPÄÄTÖKSIÄ

Geneettiset algoritmit ovat heuristisia globaalin optimoinnin menetelmiä, joilla etsitään ongelmaan globaalin optimin sijasta riittävän hyvää käypää ratkaisua. Ne löytävät kaikenlaisiin ongelmiin hyviä ratkaisuja ja niitä on käytetty menestyksekkäästi ongelmissa, joiden ratkaiseminen muilla menetelmillä on vaikeaa tai jopa mahdotonta. Geneettiset algoritmit käyttävät uusien ratkaisuvaihtoehtojen luomisessa joitakin luonnon evoluution mekanismeja: valintaa, risteytystä ja mutaatiota. Menetelmän periaatteet ovat yksinkertaiset ja niitä käyttäen voidaan ohjelmoida ratkaisualgoritmeja monenlaisiin tehtäviin. Geneettisten algoritmien etu on niiden tuottamien ratkaisujen laadukkuus. Haittoja puolestaan ovat menetelmän hitaus ja raskaus. Geneettisiä algoritmeja kannattaakin käyttää vain silloin, kun yksinkertaisemmista metodeista ei ole hyötyä.

Tässä tutkielmassa käsitelty raakapuun laivausongelma on huomattavasti monimutkaisempi ja strukturoidumpi kuin geneettisillä algoritmeilla tyypillisesti ratkaistut ongelmat. Alusten optimaalinen allokointi geneettisellä algoritmilla osoittautuikin melko vaikeaksi. Ongelmia tuottivat sopivan tietorakenteen ja operaattoreiden kehittäminen sekä parametrien arvojen asettaminen. Tehtävän rajoitteet ja datan määrä johtivat suureen tietorakenteeseen, jonka käsittely on työlästä. Tietorakenteesta ja rajoitteista johtuen ei-käypiä ratkaisuja syntyy helposti, minkä vuoksi operaattorit pyrittiin rakentamaan siten, että ne mahdollisimman hyvin vastaisivat tietorakennetta ja edesauttaisivat käypien ratkaisujen muodostumista. Koska ongelma ei ole separoituva, ei geneettiselle algoritmille voitu muodostaa selkeitä rakennuspalikoita, joita erikseen optimoimalla ja sopivasti yhdistelemällä voitaisiin päätyä globaaliin optimiin.

Vaikka geneettinen algoritmi juuttuu lokaaliin optimiin, on menetelmällä saatu tulos vähintään kohtuullinen. Kokonaiskustannus jäi selvästi suuremmaksi kuin GAMS:llä saatu vertailutulos, mutta on syytä pitää mielessä, että GAMS:n antama ratkaisu ei ole käypä. Geneettisen algoritmin suoritusaika on huomattavasti pidempi kuin GAMS-

ohjelman, mutta optimointiin kuluva aika on kohtuullinen ottaen huomioon ongelman monimutkaisuuden. Algoritmin antama ratkaisu huomioi GAMS-mallia paremmin reaalielämän rajoitteet. Tosin geneettisistä algoritmeista juuri tämän vuoksi voisi olla enemmän hyötyä operatiivisen toiminnan suunnittelussa, kun taas tässä tutkimuksessa käytetty puunkuljetusongelma on luonteeltaan strateginen.

C-ohjelmointikieli soveltuu erittäin hyvin geneettisten algoritmien kirjoittamiseen ja sitä onkin käytetty useissa alan sovelluksissa. Se on teknisesti hyvä valinta, koska sen laskennallinen tehokkuus riittää vaikeidenkin ongelmien ratkaisemiseen kohtuullisessa ajassa. Lisäksi C-kieli on laajassa käytössä monilla aloilla, joten esimerkkejä, oppaita ja funktiokirjastoja on saatavilla runsaasti.

Näyttäisi siltä, että geneettinen algoritmi ei ole paras mahdollinen ratkaisukeino tässä tutkielmassa käsitellyn puunkuljetusongelman tyyppisiin vahvasti strukturoituihin, separoitumattomiin tehtäviin. Geneettisellä algoritmilla saadaan suhteellisen laadukkaita ratkaisuja ja algoritmista voidaan tehdä mielivaltaisen realistinen, mutta tästä saatava hyöty ei korvaa menetelmän soveltamisesta aiheutuvia kustannuksia etenkään, jos ongelma on luonteeltaan strateginen. Jos kuitenkin tuloksen realistisuus on tärkeä tekijä, on geneettinen algoritmi varmempi valinta kuin valmiit optimointiohjelmistot.

Ratkaisukeinoa valittaessa kannattaa huomioida, että geneettisen algoritmin soveltaminen monimutkaiseen ongelmaan vaatii sekä ongelman että geneettisten algoritmien toimintaperiaatteiden syvällistä ymmärtämistä. Algoritmi joudutaan aina räätälöimään kulloinkin käsillä olevaan ongelmaan, mikä voi olla hyvinkin työlästä ja aikaavievää. On toivottavaa, että alan vilkas tutkimus johtaa tulevaisuudessa sellaisten menetelmien kehittämiseen, joiden avulla geneettisten algoritmien soveltaminen monimutkaisiin reaalielämän ongelmiin helpottuu.

LÄHTEET

Alander, Jarmo T. 1998. *Geneettisten algoritmien mahdollisuudet*. Teknologiakatsaus 59/98.

Fernández, F. & Tomassini, M. & Vanneschi, L. 2002. An Empirical Study of Multipopulation Genetic Programming. *Genetic Programming and Evolvable Machines*, 4, 2003, 21-51. Kluwer Academic Publishers.

Floudas, A. & Pardalos, P. (toim.) 1998. *Encyclopedia of Optimization*. Kluwer Academic Publishers.

Fogel, D. B. 1999. An Introduction to Evolutionary Computation and Some Applications. *Evolutionary Algorithms in Engineering and Computer Science*, 23-41. K. Miettinen, M. Mäkelä, P. Neittaanmäki, J. Périaux (toim.). Wiley, New York.

The GAMS system 2004.
http://www.gams.com/docs/intro.htm Kesäkuu 2004.

Goldberg, David E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Publishing Company, Inc.

Grefenstette, J. & Gopal, R. & Rosmaita, R. & Gucht., D. 1985. Genetic Algorithms for the Traveling Salesman Problem. *Proceedings of the Second International Conference on Genetic Algorithms.* Lawrence Eribaum Associates, Mahwah, NJ.

Holland, J. H. 1975. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press. Ann Arbor, Michigan.

Kernighan, B. W. & Ritchie, D. M. 1988. *The C Programming Language, 2nd Edition*. Prentice Hall, Inc.

Lallukka, Saara 2003. *Mixed Integer Programming in the Selection of Roundwood Vessels, Case Thomesto*. Julkaisematon pro gradu –tutkielma. Helsingin kauppakorkeakoulu.

Mitchell, Melanie 1996. *An Introduction to Genetic Algorithms.* The MIT Press, Massachusetts.

Reeves, C. R. (toim.) 1995. *Modern Heuristic Techniques for Combinatorial Problems*. McGraw-Hill, Lontoo.

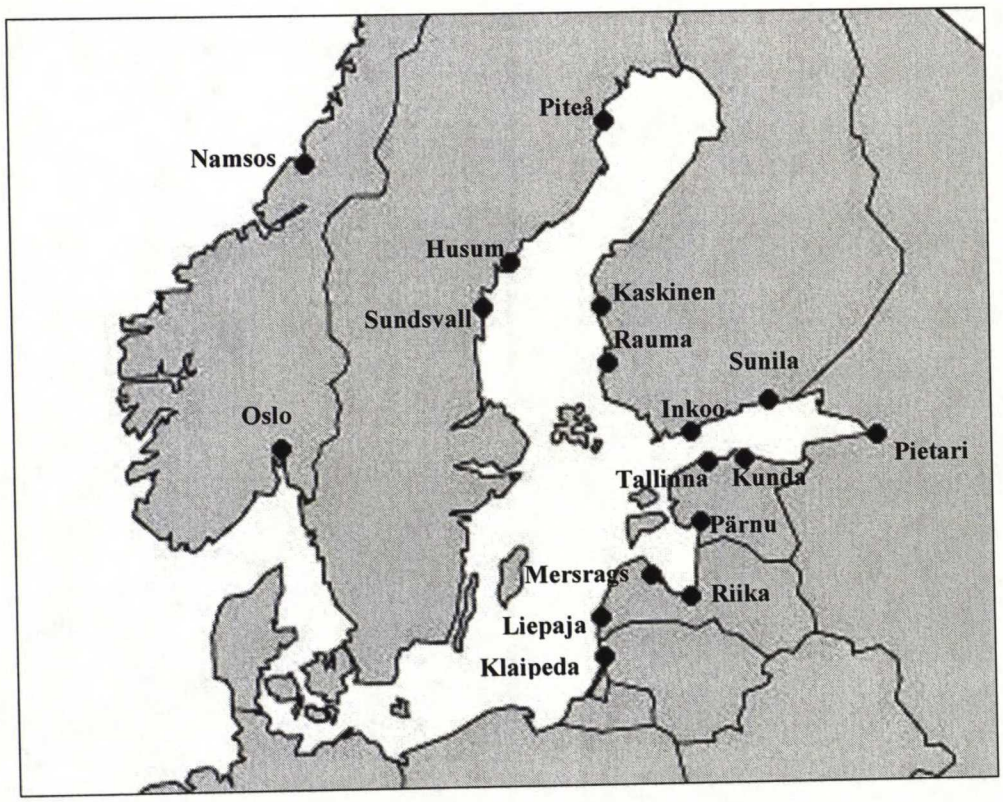
Salo, Seppo 1996. *Matemaattinen optimointi*. Helsingin kauppakorkeakoulun julkaisuja, Opetusmonisteita, O-250.

Siltanen, Heikki 1998. *Geneettisen algoritmin soveltaminen stokastiseen optimointiin.* Pro gradu –tutkielma. Helsingin kauppakorkeakoulu.

Simula 1997. *Johdatus evoluutiolaskentaan ja geneettisiin algoritmeihin.* www.lce.hut.fi/teaching/S-114.240/k97/ga/ Kesäkuu 2004.

Stopford, M. 1997. *Maritime Economics*. London and New York: Routledge.

LIITE 1. Puunkuljetusongelmaan liittyvät satamat



LIITE 2. **Alusten tekniset tiedot**

ALUSTYYPPI	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
Nopeus (solmua)	13,5	11	12	11	13	12	13	12	11	12	13	12	11
(km/h)	25,0	20,4	22,2	20,4	24,1	22,2	24,1	22,2	20,4	22,2	24,1	22,2	20,4
Polttoaineen kulutus													
- merellä (t/päivä)	8	10	8	6	8	6	11	10	8	10	12	11	6
- satamassa (t/päivä)	1	1	1	1	1	1	1	1	0,8	0,5	1	1	0,5
Polttoainelaatu	MGO	MGO	MGO	MGO	MGO	MGO	IFO 180, MGO	MGO	MGO	MGO	IFO 180, MGO	MGO	MGO
Kapasiteetti (m^3)													
- koivukuitu	3,800	4,250	3,300	2,350	3,400	2,600	5,500	5,100	2,100	4,400	4,200	5,500	1,980
- havupuukuitu	4,408	4,930	3,828	2,726	3,944	3,016	6,380	5,916	2,436	5,104	4,872	6,380	2,297
- tukki	4,672	5,226	4,058	2,890	4,181	3,197	6,763	6,271	2,582	5,410	5,164	6,763	2,435
- puuhake	3,200	3,200	3,000	1,850	3,000	2,400	4,200	4,200	2,500	3,200	3,000	4,200	2,300
- mekaaninen haapa	4,408	4,930	3,828	2,726	3,944	3,016	6,380	5,916	2,436	5,104	4,872	6,380	2,297
- haapakuitu	4,408	4,930	3,828	2,726	3,944	3,016	6,380	5,916	2,436	5,104	4,872	6,380	2,297
- kansilasti	1,000	1,300	1,000	500	700	700	1,800	1,800	500	1,500	1,100	1,800	150
Kiinteät kust. (€/päivä)	3,800	3,800	3,800	3,000	4,000	2,800	4,700	4,250	2,800	3,800	4,000	4,400	2,600
Vakuutuskust. (€/matka)	300	300	300	250	300	250	300	300	300	300	300	300	250

LIITE 3. **Suomen, Ruotsin ja Norjan satamakustannukset**
(€, SEK tai NOK/käynti satamassa)

ALUSTYYPPI	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
SUOMI (€)													
Inkoo	2 761	2 757	2 781	2 393	2 331	2 337	2 911	2 863	2 331	2 747	2 940	2 911	2 326
Kaskinen	2 429	2 426	2 460	2 040	1 947	1 955	2 656	2 584	1 947	2 410	2 699	2 656	1 939
Rauma	2 179	2 177	2 204	1 865	1 792	1 797	2 362	2 304	1 792	2 163	2 397	2 362	1 784
Sunila	3 870	3 867	3 898	3 563	3 499	3 504	4 185	4 006	3 499	3 852	4 223	4 185	3 493
RUOTSI (SEK)													
Husum	19 012	19 002	18 032	14 476	18 262	14 636	21 352	19 352	15 276	19 112	20 892	22 592	15 406
Piteå	29 200	29 200	27 700	22 100	28 050	22 350	32 050	29 400	23 300	28 350	29 650	34 950	23 700
Sundsvall	23 494	23 478	21 965	17 030	22 324	17 279	25 936	25 312	18 558	23 650	25 530	27 870	18 916
NORJA (NOK)													
Namsos	42 000	42 000	44 200	35 200	39 300	36 200	48 000	48 000	35 200	45 500	46 800	50 600	33 000
Oslo	24 000	24 000	24 900	22 100	23 300	23 000	27 300	27 300	23 000	24 900	26 000	30 000	21 300

LIITE 4. **Venäjän, Viron, Latvian ja Liettuan satamakustannukset**
(€ tai USD/käynti satamassa)

ALUSTYYPPI	k1	k2	k3	k4	k5	k6	k7	k8	k9	k10	k11	k12	k13
VIRO (€)													
Kunda	4 000	5 300	4 700	3 400	4 500	3 600	5 700	5 500	4 000	5 450	5 600	6 200	4 000
Pärnu	4 700	6 100	5 400	4 100	5 500	4 100	6 400	6 200	4 400	6 150	6 250	7 000	4 600
Tallinn	4 000	5 300	4 700	3 400	4 500	3 600	5 700	5 500	4 000	5 450	5 600	6 200	4 000
LATVIA (USD)													
Liepaja	4 268	4 069	5 015	3 602	5 149	3 696	5 992	5 732	4 131	5 708	5 836	6 734	4 282
Mersrags	3 893	3 732	4 561	3 314	4 681	3 399	5 442	5 228	3 793	5 207	5 303	6 107	3 911
Riga	4 272	4 085	5 050	3 598	3 538	3 697	6 046	5 796	4 147	5 771	5 883	6 820	4 285
LIETTUA (€)													
Klaipeda	6 164	6 151	5 145	3 678	5 623	3 795	6 627	6 365	4 176	6 259	6 401	7 437	4 329
VENÄJÄ (USD)													
Pietari 1 ja 2	7 329	8 252	7 370	5 442	8 069	5 209	9 855	9 855	6 580	8 562	8 796	10 131	6 467

LIITE 5. Kysyntä satamittain ja puulaaduittain (1000 m^3)

	Koivu-kuitu	Havupuu-kuitu	Tukki	Puuhake	Mek. haapa	Haapa-kuitu	
Inkoo	-	-	-	-	118	-	
Kaskinen	387	-	39	-	-	65	
Rauma	-	76	144	164	-	-	
Sunila	-	145	-	300	-	-	
SUOMI yht.	**387**	**221**	**183**	**464**	**118**	**65**	**1 438**
Husum	549	492	-	223	-	61	
Piteå	47	-	30	-	-	-	
Sundsvall	9	246	17	-	-	-	
RUOTSI yht.	**605**	**738**	**47**	**223**	**-**	**61**	**1 674**
Namsos	-	-	60	-	-	-	
Oslo	-	-	48	-	-	-	
NORJA yht.	**-**	**-**	**108**	**-**	**-**	**-**	**108**

KYSYNTÄ YHT.	**992**	**959**	**338**	**687**	**118**	**126**	**3 220**

LIITE 6. Tarjonta satamittain ja puulaaduittain (1000 m^3)

	Koivu-kuitu	Havupuu-kuitu	Tukki	Puuhake	Mek. haapa	Haapa-kuitu	
Kunda	114,5	58,7	-	135,7	30,2	18	
Pärnu	120,8	61,9	-	135,7	31,9	19	
Tallinn	82,7	42,4	-	135,7	21,8	13	
VIRO yht.	**318**	**163**	**-**	**407**	**84**	**50**	**1 022**
Liepaja	129,1	143,8	-	-	12	17,6	
Mersrags	59,3	131,3	-	98,1	-	11	
Riga	160,5	350	-	167	12	26,4	
LATVIA yht.	**349**	**625**	**-**	**265**	**24**	**55**	**1 318**
Klaipeda	118	139	36	61	17	32	
LIETTUA yht.	**118**	**139**	**36**	**61**	**17**	**32**	**403**
Pietari 1	151,1	53	318	-	-	-	
Pietari 2	114	15	-	-	-	-	
VENÄJÄ yht.	**265**	**68**	**318**	**-**	**-**	**-**	**651**

TARJONTA YHT.	**1 050**	**995**	**354**	**733**	**125**	**137**	**3 394**

LIITE 7. Satamien väliset etäisyydet (km)

	Pärnu	Tallinn	Kunda	Riga	Liepaja	Mersrags	Klaipeda	Pietari
Inkoo	475	80	129	475	426	426	509	338
Kaskinen	663	536	624	663	605	615	674	797
Rauma	581	459	547	581	529	533	599	715
Sunila	600	159	80	600	555	552	634	171
Husum	742	613	702	742	678	694	753	879
Piteå	949	826	914	949	887	901	964	1091
Sundsvall	666	538	626	666	602	618	678	803
Namsos							1722	2324
Oslo							932	1553

LIITE 8. Yksilön kromosomit

LIITE 9. Paras ratkaisu geneettisellä algoritmilla

Raakapuu	Kysyntä-satama	Tarjonta-satama	alus												
			k1a	k2a	k3a	k3b	k6a	k7a	k8a	k8b	k9a	k11a	k11b	k12a	k13a
Koivukuitu	Kaskinen	Pärnu								73 800					
		Mersrags								59 300					
		Klaipeda										46 800			
		Pietari 1								151 100					
		Pietari 2										56 000			
	Husum	Tallinna		82 700									114 500		
		Kunda												160 500	
		Riika											120 100		
		Liepaja													
		Klaipeda	71 200												
	Piteå	Pärnu		47 000											9 000
	Sundsvall	Pietari 2							28 242						
Havupuu-kuitu	Rauma	Pärnu							42 042						
		Mersrags										5 716			
		Pietari 1										10 486			
	Sunila	Pärnu								27 140					
		Tallinna								27 140	17 515				
		Kunda								5 720					
		Klaipeda					43 499								
		Pietari 1					13 500								
		Pietari 2													
	Husum	Pärnu		16 737									11 700	2 100	
		Tallinna													53 000
		Riika	52 187											143 800	
		Liepaja													88 364
		Mersrags						118 100							
		Klaipeda		6 012				244 500							
	Sundsvall	Riika												1 500	
		Pietari 1													

Raakapuu	Kysyntä-satama	Tarjonta-satama	alus												
			k1a	k2a	k3a	k3b	k6a	k7a	k8a	k8b	k9a	k11a	k11b	k12a	k13a
Puuhake	Rauma	Pärnu							65 900						
		Mersrags							98 100						
	Sunila	Pärnu										21 450			
		Tallinna									128 550				
		Kunda									128 550				
		Klaipeda									21 450				
	Husum	Pärnu		41 200											
		Riika	167 000												
		Klaipeda		14 800											
Tukki	Kaskinen	Pietari 1										39 000			
	Rauma	Pietari 1			144 000										
	Piteå	Pietari 1				30 000									
	Sundsvall	Pietari 1		17 000											
	Namsos	Pietari 1				60 000									
	Oslo	Klaipeda					36 000								
		Pietari 1					12 000								
Mekaaninen	Inkoo	Pärnu			31 900										
haapa		Tallinna							21 800						
		Kunda					30 200								
		Riika									5 100				
		Liepaja			12 000										
		Klaipeda					17 000								
Haapakuitu	Kaskinen	Pärnu					19 000								
		Riika					26 400								
		Klaipeda										19 600			
	Husum	Tallinna											13 000		
		Kunda				18 000									
		Liepaja				17 600									
		Mersrags		11 000											
		Klaipeda												1 400	

LIITE 10. Optimiratkaisu GAMS:llä

Raakapuu	Kysyntä-satama	Tarjonta-satama	alus											
			k1a	k2a	k6a	k6b	k7a	k10a	k10b	k10c	k12a	k12b	k12c
Koivukuitu	Kaskinen	Pärnu									10 400		
		Tallinna									82 700		
		Kunda									114 500		
		Liepaja						26 864				93 236	
		Mersrags						59 300					88 776
	Husum	Pärnu											
		Riika				135 124							
		Klaipeda				118 000							111 198
		Pietari 1								39 902			
		Pietari 2								56 000			
	Piteå	Pärnu					21 624						
		Riika					25 376						
	Sundsvall	Liepaja		9 000									
Havupuu-kuitu	Rauma	Pärnu										27 620	
		Mersrags						27 648					
		Klaipeda				11 527						9 206	
	Sunila	Tallinna			34 933			7 467					
		Kunda			26 725			31 975					
		Mersrags			8 047	294							
		Pietari 1						35 558					
	Husum	Pärnu							19 885				10 737
		Mersrags	9 671						85 640				
		Riika	89 371	39 588					118 840				
		Klaipeda								118 268			
	Sundsvall	Liepaja		143 800									
		Riika		102 200									

Raakapuu	Kysyntä-satama	Tarjonta-satama	alus										
			k1a	k2a	k6a	k6b	k7a	k10a	k10b	k10c	k12a	k12b	k12c
Puuhake	Rauma	Pärnu										64 446	
		Mersrags						38 554					
		Klaipeda				39 520						21 480	
	Sunila	Tallinna			119 771			15 929					
		Kunda			91 629			44 071					
		Mersrags			27 591	1 009							25 054
	Husum	Pärnu											
		Mersrags	30 946									39 000	
		Riika	167 000								91 912	52 088	
Tukki	Kaskinen	Pietari 1							18 437	11 563			
	Rauma	Pietari 1							17 000				
	Piteå	Pietari 1											
	Sundsvall	Pietari 1				48 000	12 000						
	Namsos	Pietari 1				36 000							9 600
	Oslo	Klaipeda				2 400							
		Pietari 1						18 417			6 583		
Mekaaninen haapa	Inkoo	Pärnu			21 800								
		Tallinna			30 200								
		Kunda			12 000								
		Liepaja			1 600			10 400					
		Riika			17 000								
		Klaipeda						13 000					
Haapakuitu	Kaskinen	Tallinna									14 600		
		Kunda						11 000					
		Mersrags						26 400					
		Riika								8 000			
	Husum	Pärnu											
		Kunda	3 400										
		Liepaja		17 600					32 000				
		Klaipeda											